SIGMOD Record – September 2005 – Volume 34 Issue 3 –

[**Editor's note**: With the exception of the last pages –which would be the back cover of the printed issue– that are not included in this file, it has the same contents as the printed edition. All the articles are also available individually online and have been put together here for convenience only.]

SIGMOD Officers, Committees, and Awardees

Chair
Raghu Ramakrishnan
Department of Computer Sciences
University of Wisconsin-Madison
1210 West Dayton Street
Madison, WI 53706-1685
USA
raghu@cs.wisc.edu

Vice-Chair
Yannis Ioannidis
University Of Athens
Department of Informatics & Telecom
Panepistimioupolis, Informatics Bldngs
157 84 Ilissia, Athens
HELLAS
yannis@di.uoa.gr

Secretary/Treasurer
Mary Fernández
ATT Labs - Research
180 Park Ave., Bldg 103, E277
Florham Park, NJ 07932-0971
USA
mff@research.att.com

Information Director: Alexandros Labrinidis, University of Pittsburgh, labrinid@cs.pitt.edu.

Associate Information Directors: Manfred Jeusfeld, Dongwon Lee, Michael Ley, Frank Neven, Altigran Soares da Silva, Jun Yang.

Advisory Board: Richard Snodgrass (Chair), University of Arizona, rts@cs.arizona.edu, H.V. Jagadish, John Mylopoulos, David DeWitt, Jim Gray, Hank Korth, Mike Franklin, Patrick Valduriez, Timos Sellis, S. Sudarshan.

SIGMOD Conference Coordinator: Jianwen Su, UC Santa Barbara, su@cs.ucsb.edu

SIGMOD Workshops Coordinator: Laurent Amsaleg, IRISA Lab, Laurent.Amsaleg@irisa.fr

Industrial Advisory Board: Daniel Barbará (Chair), George Mason Univ., dbarbara@isse.gmu.edu, José Blakeley, Paul G. Brown, Umeshwar Dayal, Mark Graves, Ashish Gupta, Hank Korth, Nelson M. Mattos, Marie-Anne Neimat, Douglas Voss.

SIGMOD Record Editorial Board: Mario A. Nascimento (Editor), University of Alberta, mn@cs.ualberta.ca, José Blakeley, Ugur Cetintemel, Brian Cooper, Andrew Eisenberg, Leonid Libkin, Alexandros Labrinidis, Jim Melton, Len Seligman, Jignesh Patel, Ken Ross, Marianne Winslett.

SIGMOD Anthology Editorial Board: Curtis Dyreson (Editor), Washington State University, cdyreson@eecs.wsu.edu, Nick Kline, Joseph Albert, Stefano Ceri, David Lomet.

SIGMOD DiSC Editorial Board: Shahram Ghandeharizadeh (Editor), USC, shahram@pollux.usc.edu, A. Ailamaki, W. Aref, V. Atluri, R. Barga, K. Boehm, K.S. Candan, Z. Chen, B. Cooper, J. Eder, V. Ganti, J. Goldstein, G. Golovchinsky, Z. Ives, H-A. Jacobsen, V. Kalogeraki, S.H. Kim, L.V.S. Lakshmanan, D. Lopresti, M. Mattoso, S. Mehrotra, R. Miller, B. Moon, V. Oria, G. Ozsoyoglu, J. Pei, A. Picariello, F. Sadri, J. Shanmugasundaram, J. Srivastava, K. Tanaka, W. Tavanapong, V. Tsotras, M. Zaki, R. Zimmermann.

SIGMOD Digital Review Editorial Board: H. V. Jagadish (Editor), Univ. of Michigan, jag@eecs.umich.edu, Alon Halevy, Michael Ley, Yannis Papakonstantinou, Nandit Soparkar.

Sister Society Liaisons: Stefano Ceri (VLDB Foundation and EDBT Endowment), Hongjun Lu (SIGKDD and CCFDBS), Yannis Ioannidis (IEEE TCDE), Serge Abiteboul (PODS and ICDT Council).

Latin American Liaison Commitee: Claudia M. Bauzer Medeiros (Chair), University of Campinas, cmbm@ic.unicamp.br Alfonso Aguirre, Leopoldo Bertossi, Alberto Laender, Sergio Lifschitz, Marta Mattoso, Gustavo Rossi.

Awards Committee: Moshe Y. Vardi (Chair), Rice University, vardi@cs.rice.edu. Rudolf Bayer, Masaru Kitsuregawa, Z. Meral Ozsoyoglu, Pat Selinger, Michael Stonebraker.

Award Recipients:

Innovation Award: Michael Stonebraker, Jim Gray, Philip Bernstein, David DeWitt, C. Mohan, David Maier, Serge Abiteboul, Hector Garcia-Molina, Rakesh Agrawal, Rudolf Bayer, Patricia Selinger, Don Chamberlin, Ronald Fagin.

Contributions Award: Maria Zemankova, Gio Wiederhold, Yahiko Kambayashi, Jeffrey Ullman, Avi Silberschatz, Won Kim, Raghu Ramakrishnan, Laura Haas, Michael Carey, Daniel Rosenkrantz, Richard Snodgrass, Michael Ley, Surajit Chaudhuri.

Editor's Notes

Dear Colleagues,

This is the first issue with the newly elected officers for *SIGMOD*: Raghu Ramakrishnan (Chair), Yannis Ioannidis (Vice-chair) and Mary Fernández (Secretary/Treasurer). I am confident I can speak for all members when I bid a warm welcome and wish them a productive mandate. Likewise, I would also like to thank the other candidates, Ahmed Elmagarmid, Krithi Ramamritham and Louiqa Rashid, for putting forward their names. As others have mentioned it before, is good to see that *SIGMOD* can count on good people willing to keep its well-established tradition.

You will probably notice that this issue is missing the traditional Chair's Message. The reason for that is simple: timing. Even though this is September's issue, it is closed and sent to ACM's Headquarters late in July. Raghu took over shortly before I had to close this issue, and amid all the transition work, his message could not be prepared in time; hence this issue will not feature the Chair's Message. However, in a quick email exchange he asked me to relay the following message: "*I'm glad to have the opportunity to be chair and to work with Mary and Yannis, and we're working to understand the issues involved in our new offices.*" We all look forward for his first message in December's issue. The new officers are also in the process of updating the volunteer committees as well, so the listing you see on this issue is likely to change for the next ones.

This issue features a special section on Scientific Workflows guest-edited by Bertram Ludäscher (Univ. of California, Davis) and Carole Goble (Univ. of Manchester). As you can see from their introduction to the special section, they were able to put good together a good set of papers covering several aspects of this new area. I hope to be able to have similar guest-edited special section once or twice a year. If you have an idea about one and are willing to work on it, please contact me.

On a sad note, Alberto Mendelzon passed away last June. I think that the following, excerpted from a statement[1] by his colleagues in the Toronto Database Group, summarizes everyone's feelings well: "*Alberto was a great intellect, a charming friend, and an inspirational mentor. The citation for his recent election into the Royal Society of Canada (the Canadian National Academy) gives a glimpse into his intellectual legacy, but says nothing of the great man who will be sorely missed. His passing leaves a hole in our community and in our hearts.*" As I type this note, his colleagues in Toronto are working on a tribute that should appear in December's issue.

Finally, the *Record* does have a backlog of papers both to be printed and to be reviewed. While it is nice to have a larger number of papers being submitted, this puts further stress on the reviewing process (which I mentioned on June's issue). I assure you that all submitted papers are given fair attention, which, sometimes, means a longer review time. As well, all those accepted will be published as soon as possible. I wish all could be published as they are accepted, but I have to work within a pre-determined page budget, which means all papers have to wait for its turn "in line". I am sure you all understand (though I also understand you may not like it when it is *your* paper that is delayed, but *c'est la vie*).

I hope you enjoy this issue and that you have had a good Summer time (or Winter time depending on which hemisphere you are).

Mario Nascimento, Editor.
July 2005

[1] `http://www.sigmod.org/NEWS/05/alberto.16june2005.html`

Guest Editors' Introduction to the Special Section on Scientific Workflows

Bertram Ludäscher
Dept. of Computer Science & Genome Center
University of California, Davis
1 Shields Ave, Davis, CA 95616, USA
ludaesch@ucdavis.edu

Carole Goble
School of Computer Science
The University of Manchester
Manchester, M13 9PL, UK
carole@cs.man.ac.uk

1. INTRODUCTION

Business-oriented workflows have been studied since the 70's under various names (office automation, workflow management, business process management) and by different communities, including the database community. Much basic and applied research has been conducted over the years, e.g. theoretical studies of workflow languages and models (based on Petri-nets or process calculi), their properties, transactional behavior, etc.

Recently, and largely unnoticed by the database community, *scientific workflows* have gained momentum due to their central role in e-Science and cyberinfrastructure applications, i.e., where scientists need to "glue" together data management, analysis, simulation, and visualization services over often voluminous and (structurally and semantically) complex, distributed scientific data and services. While sharing commonalities with their business workflow relatives, scientific workflows often pose different challenges. For example, scientific workflows are typically data-centric, dataflow-oriented "analysis pipelines" (as opposed to task-centric and control-flow oriented business workflows) and can be very computationally expensive (often requiring parallel and/or Grid computing capabilities).

Another characteristic is that scientific workflows are often more metadata and annotation-intensive, since repurposing of a scientific data product in another scientist's study requires detailed (and preferably machine-processable) context and data provenance information. Finally, scientists typically are rather individualistic and are more likely to create their own "knowledge discovery workflows", whereas in business, users are commonly restricted to using carefully designed and predetermined automation workflows in a constrained way.

Scientific workflow *systems* are related to (and can have features of) mathematical problem solving environments [1], LIMS (Laboratory Information Management Systems), dataflow visualization systems (AVS, IBM's OpenDX, SciRUN, etc.), and distributed (Grid) scheduling and execution environments. Users of scientific workflow systems range from bench scientists to computational scientists and of course include the new breed of "hybrid" e-scientists. Scientific workflows are useful to capture, document, archive, share, execute, and reproduce scientific data analysis pipelines from all disciplines (e.g., biology, medicine, ecology, chemistry, physics, geosciences, and astronomy). Clearly, different disciplines and subdisciplines can have different requirements and characteristics w.r.t. data volume, (structural and semantic) heterogeneity, computational complexity, etc.

Grid computing (now largely web service based) has stimulated workflow developments, from the orchestration of long running applications to the scheduling of job submissions to marshalled compute resources. Many scientific workflow systems can execute remote web services and local tools (e.g., via a command-line interface).

2. SPECIAL SECTION OVERVIEW

With this special section we aim at providing a glimpse of a number of research and development activities and technical challenges in scientific workflows. Due to space limitations, we can only provide a very limited snapshot of ongoing work. Nevertheless, we hope that this special section can serve as a first sample of the range of issues that define the current state-of-the-art in scientific workflows and that provide a starting point for further research and contributions by the database community.

The call for papers attracted 31 submissions, indicating the large interest in the topic. Based on the peer reviews by about thirty external reviewers, 9 papers were accepted. Several of the papers use case studies from the life sciences: two papers use applications in biomedical image analysis, and two others use bioinformatics and phylogenetics examples. The geosciences are also represented. These disciplines are characterized by large, distributed and heterogeneous data sets, which are subject to change and regular re-interpretation, and need to be combined and processed in differing and non-prescriptive ways by third party scientists.

Scientific Workflow Systems. There is a plethora of scientific workflow environments covering a range of scientific disciplines. YU and BUYYA's *"Taxonomy of Scientific Workflow Systems for Grid Computing"* sets the scene by briefly characterizing and classifying various approaches for building and executing workflows on the Grid. A comprehensive scientific workflow system has demanding execution requirements. They should be able to schedule workflow tasks (typically in a distributed/Grid environment), monitor and control execution, allow on-the-fly visualization and computational steering, facilitate "pause and rerun", gracefully manage failure, and support various static and dynamic analysis and optimization techniques.

Metadata for Workflow Reuse and Provenance. Scientific workflows are pivotal knowledge components in e-Science. The scientific protocol encapsulated by the workflow provides a context and history for its products that

enables their interpretation. Data provenance is such a critical component of scientific workflows, that the *"Survey of Data Provenance in e-Science"* by SIMMHAN, PLALE and GANNON is a welcome summarization of the key research efforts and open challenges.

A workflow is itself is know-how about a scientific method that can be shared and reused, or act as a template for new versions of workflows. By reusing workflows we can spread best practice, avoid wasteful duplicated effort, and foster scientific collaboration. Along with shared data warehouses and service registries, we envision shared catalogues of workflows indexed by metadata, as do MEDERIOS ET AL in *"WOODSS and the Web: Annotating and Reusing Scientific Workflows"*. This presents challenges of how to describe and query workflows, understanding models of reuse, and presenting the workflows in terms of a user model rather than a delivery paradigm.

In *"Simplifying Construction of Complex Workflows for Non-Expert Users of the Southern California Earthquake Centre Community Modelling Environment"* MAECHLING ET AL pick up this theme by returning the scientist at the center of a real scientific workflow application. Workflow templates are shared and reused by scientists; metadata is used to intelligently guide scientists to build and refine their own workflows.

Workflow Support for Data Collections. As scientific data analysis is the main use of workflows, it is becoming apparent that large-scale data-intensive workflows will dominate e-Science. Two papers take the management of data collections in workflow as their theme, whilst two more take a more conventional database line, arguing that database technologies can support workflow environments.

When constructing workflows that operate on large and complex datasets, the ability to describe and introspect on the types of both datasets and workflow components is invaluable – for type checking and iteration over collections, for example. If the datasets were described using clearly defined and shared metadata, and stored in well-organized databases, then this would be straightforward. However, the real world is not like this. Datasets are commonly files, and metadata is encoded in directory and file names, employed in ad-hoc ways. The physical manifestation of the dataset is conflated with its logical structure.

In *"A Notation and System for Expressing and Executing Cleanly Typed Workflows on Messy Scientific Data"*, ZHAO ET AL present a typed workflow notation and system that allows workflows to be expressed in terms of abstract XML data types that are then executed over diverse physical representations, decoupling the physical and logical descriptions without forcing change in the datasets themselves.

MCPHILLIPS and BOWERS in *"An Approach for Pipelining Nested Collections in Scientific Workflows"* take up the theme of appropriate approaches for workflow execution over large-scale nested data collections. Their framework illustrates a new scientific workflow programming paradigm, emphasizing extensibility through collection-aware actors, concurrent operations, on the fly component customization and exception management.

Database Support for Workflow Execution. Several works focus on database support for scientific workflows. SHANNON ET AL pick up on the prevalence of XML for describing and representing datasets, that there should be *"XML Database Support for Distributed Execution of Data-intensive Scientific Workflows"*. They use the Mobius framework for on demand creation and federation of XML databases and DataCutter for streaming data between processes.

SHANKER ET AL go further by arguing in *"Integrating Databases and Workflow Systems"* that workflow execution and data management are so co-dependent that this calls for a workflow modelling language that tightly integrates workflow management systems and database management systems. Rather than a process dominated viewpoint, where data is a product of a workflow engine, they see workflow execution as a means of generating data products as an extension of SQL, putting the database at the center rather than the workflow execution machinery.

This resonates with the Virtual Data Language discussed by ZHAO ET AL. A more loosely coupled approach has been proposed by the OGSA-DQP project; they suggest that database queries can be workflow jobs and workflow components can be queries [2].

Workflow Scheduling. In *"Scheduling of Scientific Workflows in the ASKALON Grid Environment"*, WIECZOREK, PRODAN and FAHRINGER's paper focuses particularly on execution performance for scheduling in Grid environments, and represent a particular use of workflow, that is scheduling *job* submissions over compute resources, sometimes termed "workflow in the small". This is in contrast to workflows for orchestrating *applications*, termed "workflow in the large", such as those developed by myGrid's Taverna system [3] or Kepler [4].

Conclusion. It has been recognized by funding agencies and their respective programmes and initiatives (e.g., NIH Roadmap, NSF ITR, Cyberinfrastructure, DOE SciDAC, UK e-Science, various EU programmes, etc.) that scientific advances and discoveries are facilitated through novel IT infrastructure and tools. Scientific workflows provide the interface between scientists and this infrastructure. We think that the many and various types of technical challenges in scientific workflow modeling, design, optimization, verification etc. provide a rich playing field and great opportunity for database researchers.

Acknowledgements. We thank the SIGMOD-Record editor, Mario Nascimento, and all external reviewers of this special section for their support.

3. REFERENCES

[1] R. F. Boisvert and E. N. Houstis, editors. *Computational Science, Mathematics, and Software*. Purdue University Press, 1999.

[2] OGSA-DQP: Service Based Distributed Query Processor. `http://www.ogsadai.org.uk/dqp/`.

[3] myGrid. `http://www.mygrid.org.uk/`.

[4] Kepler. `http://kepler-project.org/`.

Integrating databases and workflow systems

Srinath Shankar Ameet Kini David J DeWitt Jeffrey Naughton

Department of Computer Sciences
University of Wisconsin
Madison, WI 53706-1685
{srinath,akini,dewitt,naughton}@cs.wisc.edu

Abstract

There has been an information explosion in fields of science such as high energy physics, astronomy, environmental sciences and biology. There is a critical need for automated systems to manage scientific applications and data. Database technology is well-suited to handle several aspects of workflow management. Contemporary workflow systems are built from multiple, separately developed components and do not exploit the full power of DBMSs in handling data of large magnitudes. We advocate a holistic view of a WFMS that includes not only workflow modeling but planning, scheduling, data management and cluster management. Thus, it is worthwhile to explore the ways in which databases can be augmented to manage workflows in addition to data. We present a language for modeling workflows that is tightly integrated with SQL. Each scientific program in a workflow is associated with an **active table** or view. The definition of data products is in relational format, and invocation of programs and querying is done in SQL. The tight coupling between workflow management and data-manipulation is an advantage for data-intensive scientific programs.

1 Introduction

Cutting edge science has been witness to an information explosion. In recent years, scientists from fields such as high energy physics, astronomy, environmental sciences and biology have been overwhelmed by the task of managing the vast quantities of data generated by their experiments. Some examples are ATLAS [7], SDSS [10], GEON [5] and BIRN [1]. In addition to the data, scientists also have to deal with a large number of specialized programs. There is a critical need for automated systems to manage scientific applications and data. Thus, the study of scientific workflows has gained importance in its own right. Several WFMSs (GriPhyn [6], Griddb [29], Zoo [26], Kepler [12]) have been proposed to provide functionality such as workflow modeling, execution, provenance, auditing and visualization. The importance of the Grid and systems like Condor [4] to workflow management has also been recognized. While database technology has been utilized to some extent in each of these systems, none of them really explore the full power of DBMSs in handling large magnitudes of data. Furthermore, contemporary WFMSs are built from multiple components, each of which performs a separate function. We believe it is a mistake to study the different aspects of workflow management in isolation. Databases should play a crucial role in the big picture, and this must motivate architectural decisions with respect to workflow modeling, planning, execution and data management.

2 The Grid - An added dimension

Recent research in scientific workflow management has focused on the Grid. Several definitions of the grid exist [16]. Theoretically, a computing grid is like a power grid. Users can 'plug into' it and avail themselves of the vast computing resources available around the world.

The most popular distributed computing system that approximates this behavior is Condor, which is used to manage clusters (or pools) of machines. Condor is an excellent way to harvest the CPU cycles of idle machines. It provides useful virtualizing services, such as migrating jobs and transferring the input and output files for a job to the right machine. Furthermore, it is easy to customize on a per-job basis. Users can specify the type of environments they need to run their jobs, and individual machines in the pool can implement policies regarding the kinds of jobs they are willing to run. Thus, it is an invaluable tool for scientists with long-running, resource-intensive jobs.

Condor has established itself as the work-horse of scientific computing. At last count, (Jul 2005) it was installed on more than 58000 machines organized across 1500 pools in universities and organizations world-wide. Condor was developed in the early '80s primarily as a cycle harvesting system. Since then a lot of the assumptions that motivated the design of Condor have changed. In the '80s computing resources were scarce and expensive. Now, large clusters of commodity machines have become affordable for almost everyone. Although the volumes of data available to and processed by typical scientific applications has increased, the cost of storage has also plummeted. In light of the advances in processor and disk technology over the last decade, we believe it is worthwhile to re-evaluate some of the design decisions made in the architecture of Condor. As the need for cycle harvesting on idle desktop machines diminishes and the need for better cluster management comes to the

fore, we feel that grid computing systems such as Condor can exploit database technology to provide improved functionality, scalability and reliability.

3 Related work

A lot of research has already been done in the field of managing *business* workflows with databases. Examples include active databases [31], enhanced datalog [15] and relational transducers [11]. Oracle 9i introduced job queues for the periodic execution of administrative and house-keeping tasks defined in the procedural PL/SQL language. Oracle 10g offers database views for monitoring the status of jobs, programs, program arguments and schedules. Solutions for business workflows have mainly concentrated on the *control* flow between processes.

Scientific workflows, on the other hand, pose an entirely new set of challenges. Scientific jobs are usually long-running and highly resource-intensive. Thus, efficient *data*-flow management is essential. In addition, scientists require sophisticated tools to query and visualize their data products. One of the first systems built to handle workflows in this domain was Zoo [26], a desktop experiment management environment. It used the object-oriented language Moose to model control flow between jobs as relationships between entities that represented jobs, input data and output data. Job invocation was handled by assigning rules on these relationships. Zoo also provided a variety of workflow auditing capabilities that allowed users to query the state of the database using the Fox query language. The Zoo system was architected with a data-centric view of workflows and was built on the Horse OODBMS.

The GriPhyn project [6] is a collaborative effort toward a standard solution for handling scientific workflows. Its components include Chimera [23], Pegasus [21] and a Replica Location Service [19]. Chimera allows users to declare programs, their logical data products and the composite workflow graph in a Virtual Data Language. Given a request for a particular virtual data product, Chimera analyzes the Virtual Data Catalog and comes up with an abstract DAG representing the sequence of operations that produce that data. Pegasus, the planner, locates physical file replicas (using RLS) and uses resource information (from the Globus Monitoring and Discovery Service [25]) to come up with a concrete plan of execution. The concrete plans produced by Pegasus are in fact Condor DAGMan [2] submit files. The DAGMan scheduler in Condor is used to execute the programs on a Condor pool. Pegasus uses artificial intelligence techniques to choose amongst multiple possible physical replicas and minimize resource usage in its planning algorithms.

The Ptolemy II [9] is based on the concept of actors, which are independent components that perform tasks such as data transformations, specific steps in an algorithm or simply opening a browser or a shell program. Actors have well-defined interfaces called ports and can be composed into scientific workflows. The idea is to promote the reuse of entities and thus make scientific workflows easy to design and more modular. Kepler [12] provides extensions to Ptolemy such as a *Webservice* actor to enable access to remote resources and *FileStager*, *FileFetcher* and *GlobusJob* actors to enable workflows to make use of the Grid. It also provides a *Director* actor, similar to Condor's DAGMan, that can be used to schedule the execution order of individual actors. The Kepler system has been used in scientific projects such as GEON, SEEK and SciDAC/SDM. While Kepler provides a useful way to model scientific workflows, it does not address the larger issues of planning workflows or providing fault-tolerance measures.

In GridDB [29], the inputs and outputs of programs are modeled as relational tables. It allows users to define programs and the relationship between their inputs and ouptuts in a functional data modeling language (FDM). Insertion of tuples in input tables triggers the execution of programs in the workflow. Programs are executed by submitting them to Condor.

Turning to workflow execution substrates, the Condor team at UW-Madison has worked on a number of projects aimed at increasing the functionality and efficiency of Condor. These include Stork [28] and DiskRouter [3] for efficient data placement, Condor-G (which integrates Condor with the Globus toolkit [25]) and Hawkeye [8] (which lets users monitor their jobs). These have been integrated with Condor in varying degrees.

Perhaps the most relevant work with respect to the execution of scientific programs was done by Gray et al. [24]. Using the cluster-finding example from the Sloan Digital Sky Survey, they demonstrated the benefits of modern DBMSs, such as using indices, parallelizing query execution and using efficient join algorithms. The performance obtained by using a database (Microsoft's SQL Server) to store and query astronomical data was orders of magnitude better than previous implementations of the algorithm. They advocate a tighter integration of computation and data, i.e. involving the DBMS in the analysis and computation of scientific applications and not relegating it to a passive store.

4 Limitations of current workflow systems

Most of the workflow management systems described above are composed of multiple components that were developed independently. Thus, inter-component communication is set up along very rigid channels. For example, consider the role of planning in a WFMS. In their

analysis of batch workloads across remote compute clusters, Bent et al. [14] showed that there are qualitative differences in the types of I/O performed by an individual job during its lifetime. *Batch* I/O refers to the input that is common to all jobs in a batch, while *pipeline* I/O is the data flow that occurs between jobs in a particular DAG. The authors recognized the importance of exploiting the sharing characteristics of batch input and providing locality of execution for programs that share pipeline I/O to minimize network traffic. Current WFMSs are not closely integrated with the user data, and are thus unable to do such detailed planning. For example Pegasus, the planning component of GriPhyN, which uses AI techniques to arrive at globally optimal solutions, doesn't have access to detailed job characteristics. GridDB currently has no way of planning workflows at all, since it is constrained by the Condor job-submission interface. Incorporating planning into such systems will require a substantial reworking of their design. While the approach pursued by Gray et al. seems the most holistic solution, it involves invoking program modules from within SQL statements (in the form of user-defined functions or stored procedures). Most databases do not directly execute binaries or interpret arbitrary code. Programmers who wish to invoke modules as part of SQL statements have to conform to strict specifications while writing and compiling them, and are usually constrained in the languages they can use (C,C++ and Java). Scientists may balk at having to rewrite their applications to allow them to be run by a database. We feel the continued usage of legacy applications in languages such as Fortran, and the importance of file-based (as opposed to database) I/O in the scientific domain may hinder the adoption of the technique espoused by Gray et al. Furthermore, rewriting workflows in SQL is not sufficient. Scientists also want ways to set up and execute workflows.

The Zoo system used objects and relationships to model workflows. Like GridDB, triggers were used to activate workflows. However, Zoo was designed as a desktop management system and not for cluster management, which is an important component of a WFMS. Zoo had no mechanisms for gathering workload characteristics, maintaining data replicas across nodes and scheduling data transfers between machines.

5 Why a database is essential

At this juncture, it is important to concretize the notion of a WFMS. How is it used, and what demands do we make of it in terms of functionality? From the user's perspective, there are three important aspects to a WFMS.

- The declaration and specification of workflows, processes and data
- The invocation of scientific workflows
- The ability to monitor workflows.

A WFMS must make it easy for a user to do the above while successfully abstracting away the following behind-the-scenes activities:

Workflow Planning – The various processes that comprise a workflow and the attendant data transfer must be planned to maximize system efficiency and throughput.

Scheduling – It is the responsibility of the WFMS to invoke individual programs and schedule data transfers according to a plan.

Data management – A WFMS must keep track of the data produced by user workflows, manage data replicas and consistency, provide data recovery in the face of failure and maintain versions of user programs and data as a workflow evolves.

Cluster management – A WFMS must monitor the state of its cluster, including network connectivity and machine parameters such as disk space and load averages. It must handle temporary and permanent machine failure in a transparent manner while distributing load evenly.

As Sections 3 and 4 show, database technology has been used only to a limited extent, mostly to store metadata and task descriptions. For instance, Griphyn's RLS simply uses a database to store its replica catalog. GridDB uses a database to store *memo tables*, which are correspondences between the inputs and outputs of a program that has completed, and *process tables* that contain state information of currently executing programs. In commercial databases, job queues have been used to schedule administrative tasks and for interprocess communication. We believe database technology has a lot more to offer the realm of scientific workflow management.

Planning – The research in [14] demonstrates the need for workflow planning if clusters are to scale to a large number of machines (about 100,000) executing huge batch workloads. No satisfying solution currently exists to this problem, and we believe it is an important one to solve in the near future. Database technology has long concerned itself with the issue of optimizing queries over distributed data sources to minimize CPU time and network and disk I/O ([22],[30]). Moreover a great deal of attention has been paid to dynamic query optimization [20], which uses feedback from currently executing queries to make plan adjustments. Thus, databases are ideally suited to planning data-intensive scientific workflows in dynamic environments.

Provenance – A good workflow management system should provide administrative assistance in addition to job scheduling services. The importance of data provenance has already been recognized in the GriPhyn and GridDB projects. Data provenance has been studied from both theoretical [17] and practical perspectives

by the database community. Data management systems such as [32] have demonstrated the ability of databases in providing real provenance services to users.

Concurrency control – As cluster and workload sizes increase, it becomes necessary to recognize and prevent interference between simultaneously executing jobs. Current WFMSs do not address this problem adequately. Databases are good at handling concurrent access at various levels of granularity, maintaining replica consistency for data objects and resolving conflicts. In addition, databases can provide different degrees of consistency for different transactions.

Recovery – Recognizing and handling multiple failure modes in a grid computing environment is very important. For example, a job must not be 'lost' or 'forgotten' when machines involved in its execution crash. Databases are good at recovering data and system state in a transparent way while minimizing the impact on currently executing processes and efficiently balancing load on the remaining machines.

Transactional semantics and persistence – The notion of a transaction is essential to workflows. The atomicity and durability of certain sequences of actions, such as moving data and jobs from one machine to another, is important. Condor's DAGMan is a custom-designed tool that allows users to specify workflow graphs that are guaranteed to be executed. Even this popular tool doesn't allow users to 'rollback' portions of a graph when their programs fail or abort jobs that were mistakenly submitted or violated certain constraints. Transactional semantics would provide greater functionality and simplify the design of WFMSs.

Querying capabilities –In addition to user data, grid compute clusters generate large amounts of operational data on a daily basis. This data includes the status of machines and jobs in the cluster, user information and file access information. For example, in Condor, such data is spread over log files across multiple machines, making it very hard to administer. In fact, a lot of it is thrown away due to archaic space management policies. The tasks of managing a pool of machines, monitoring its health and diagnosing problems could be simplified and even automated if its operational data were accessible using SQL in a uniform, declarative fashion. The support for SQL provided by a DBMS would also allow users to query the status of their personal jobs and data without impinging the privacy of other users.

To summarize, many components required of a scientific WFMS have been a part of DBMS technology for a long time. A comprehensive system that encompasses workflow planning, data and cluster management must have a database system as its core. In fact, since databases provide much of the functionality required of a WFMS, we feel it makes sense to ask – Can we augment databases to handle workflows in addition to data? A tightly integrated architecture is needed and the design of all components of a WFMS must be motivated by the use of database technology. No aspect of workflow management should be viewed in isolation from the others. For instance, a workflow modeling language that is similar to SQL could leverage database query optimization techniques for the efficient execution of data-intensive scientific jobs. In addition, in order to plan and schedule workflows, query optimizers must have access to information such as replica locations, machine availability and load in relational format. This would allow most WFMS management operations to be performed in a way analogous to traditional data management operations.

6 A modeling framework

In this section we present a workflow modeling language that tightly integrates WFMSs and DBMSs. First, we identify some of the features a workflow modeling language should possess:

Program specification – The language should let users declare programs and specify an 'invocation format' for the programs. By invocation format we mean the format in which the program and its parameters are presented to a cluster for execution. For example, the invocation format could be a shell command, or a Condor submit file.

Data specification – Users may wish to explicitly model the input and output of their programs, and a modeling language must allow users to assign a schema to this data.

Data 'transducers' – Almost all scientific programs are written to deal with data in files. Thus, a modeling language must provide for transducers that interpret data resident in flat files and provide a relational 'view' of it.

Specification of control and data flow – Users must be able to compose their programs and data into a unified workflow.

Workflow invocation – The modeling language must allow users to activate workflows in a transparent and well-defined manner.

To demonstrate our modeling language we use the ATLAS high energy physics workflow ([29]). An event generator program (*gen*) is used to feed two different simulator (*atlsim* and *atlfast*) and their output is compared (Figure 1).

The modeling language we present is based on the concept of an **active table**. Every program in a workflow is associated with an active relational table. An active table has two parts to its schema – one for the input of the program and one for its output. For example, the event generator *gen* is declared in the following way (keywords are in uppercase)

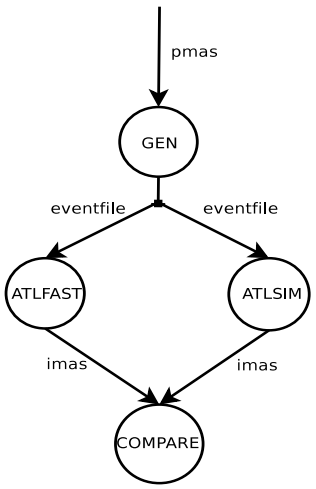


Figure 1: The simplified Atlas Workflow

```
Example I (GEN):
CREATE ACTIVE TABLE Gen
WITH PROGRAM '/path/to/gen'
INPUT (pmas INTEGER) INVOCATION (" $pmas")
OUTPUT (eventfile VARCHAR) FORMAT 'genschema.xml'
```

Program *gen* takes as input an integer *pmas* and produces a file of events (to be used by the simulators). The `INPUT` and `OUTPUT` clauses capture this information – `pmas` contains the integer parameter for *gen*, and `eventfile` contains the name of the file it produces. The schema of an active table is the concatenation of its input and output fields. Thus, the table `Gen` has the schema `Gen(pmas INTEGER,eventfile VARCHAR)`. The `INVOCATION` clause specifies that the program is invoked by issuing the command '`gen <pmas>`'. The purpose of the `FORMAT` clause is explained in the next example.

Workflows are composed by declaring active views, which are similar to active tables. Consider the program *atlfast*, which takes as input the event file produced by *gen*. The declaration of *atlfast* is as follows

```
Example II (ATLFAST):
CREATE ACTIVE VIEW Atlfast
WITH PROGRAM '/path/to/atlfast'
INPUT (eventfile varchar)
INVOCATION ("-events $eventfile")
AS SELECT Gen.eventfile FROM Gen
OUTPUT (imas integer) FORMAT 'atlschema.xml'
```

As was the case in the Example I, the relational view `Atlfast` is associated with the program *atlfast*. The active view has been defined to take as input the filename `Gen.eventfile`, which is output by the program *gen*. (It is not required that the `INPUT` field in `Atlfast` also be called `eventfile`). The table `Atlfast` has the schema `Atlfast(eventfile VARCHAR, imas INTEGER)`.

Most scientific programs process data in a filesystem. For instance, the output of the *atlfast* program is an integer which it stores in a file. To interpret data in files, the WFMS needs to know the format in which it is stored, and what the desired relational format is. The `FORMAT` clause is used to specify an XML file containing this information.

Workflow automation is accomplished in the following way – In examples I and II, `Gen` is the base table, and `Atlfast` is the derived table. The workflow is set up by populating the base table with input data. Each record of input corresponds to a separate execution of the workflow. A section of the workflow is invoked by issuing a query on the corresponding derived table. For instance, `Gen` could be populated as follows:

```
INSERT INTO Gen(pmas) VALUES (100)
INSERT INTO Gen(pmas) VALUES (200)
etc.
```

The *gen-atlfast* section of the ATLAS workflow can be explicitly started using the SQL query:

```
SELECT Atlfast.imas FROM Atlfast
```

The general idea is that a program is invoked whenever a query is issued on the `OUTPUT` fields of its active table (in the above example, the output field is `imas`). Thus, *gen* is invoked for each value of `pmas` (100,200,...), and *atlfast* is called for each event file *gen* produces. If a program fails due to an error or abnormality, the corresponding `OUTPUT` field is filled with `null` or an appropriate error value. For example, suppose the program *gen* with input 200 fails. Then the corresponding `Gen.eventfile` field is set to `null`. The exact input that caused the error can be detected by the simple query:

```
SELECT Gen.pmas FROM Gen
WHERE Gen.eventfile IS NULL
```

To complete the ATLAS example, here is the declaration of the *atlsim* program. Its input and output are similar to *atlfast*

```
Example III (ATLSIM):
CREATE ACTIVE VIEW Atlsim
WITH PROGRAM '/path/to/atlsim'
INPUT (eventfile varchar)
INVOCATION ("-events $eventfile")
AS SELECT Gen.eventfile FROM Gen
OUTPUT (imas integer) FORMAT 'atlschema.xml'
```

The comparison between the output of the two simulator programs can now be done by means of a SQL query

```
Example IV (COMPARE):
SELECT F.imas, S.imas FROM Atlfast F, Atlsim S
WHERE F.eventfile = S.eventfile
```

Thus, in this workflow modeling language, initial input data and programs are represented as active tables, and derivative programs and data as active views. For each program in the workflow, exactly one table/view has to be defined. Any section of the workflow can be explicitly invoked by issuing a SQL query on the corresponding views or tables (Such a query is the comparison query in Example IV). Workflow automation is simply an extension of active view maintenance. Since the

views are generated by the output of possibly long running programs, it might be best to materialize them instead of regenerating them on each query. This prevents unnecessary re-execution of programs. Like traditional materialized views, active views can be updated in two ways

- When base data changes (that is, input is inserted into the initial tables in the workflow), successive stages of the workflow can be invoked – the 'push' method
- When a query is issued (explicit invocation) views can be updated if necessary – the 'pull' method.

If the 'pull' mechanism is adopted, users might have to wait long periods of time for the result of a query. We feel a 'push' mechanism would increase the degree of automation the WFMS provides.

The modeling language presented above is meant for tight integration with a database. The declaration and definition of workflows is in relational format, and the invocation and querying is done in SQL. Thus, as example IV shows, data manipulation can be closely integrated with workflow execution. Moreover, the query graph that represents the output (in this case, example IV) is the same as the workflow graph. Coupled with a knowledge of the data products, this opens the door to workflow optimization using database techniques. For instance, the WFMS need not materialize the outputs of *atlfast* and *atlsim* before executing the join – a pipelined plan can be chosen in which the join is executed 'on the fly', thus improving efficiency. Architectures that simply layer a modeling framework over an execution substrate do not lend themselves to such optimizations.

7 An execution framework

We now turn to the execution substrate, with Condor as the standard. Condor consists of several daemons such as the *schedd* to which users present jobs for submission, the *startd* which is responsible for advertising machines in the pool, and the *negotiator* which matches jobs to machines. The task of managing, debugging and querying the system is a complex task involving multiple daemon log files across hundreds of machines in the pool. Recent research at UW-Madison has addressed many of the problems facing system administrators and users by providing a powerful window into the system while simultaneously laying the foundation for a more comprehensive DB-managed cluster architecture. The original Condor daemons on different machines in the pool were modified to report operational data to a central database. This data lends itself to diagnostic queries (Which machines are down and for how long? Why hasn't a job run?), as well as user queries (What is the status of my jobs? How much system run-time have I got?). Additionally, certain information that was hard to obtain previously is now easily accessible. For instance, any aggregate query on the I/O performed by a job would previously involve scanning multiple log files over all the machines the job ever executed on – now it's a simple SQL query.

The instrumentation of the Condor daemons is also an important first-step toward building a WFMS around a DBMS – the operational data collected could be used to compile statistics about jobs (such as run-time, I/O activity, disk usage), data (such as file usage, size and sharing characteristics) and machines (load average, job preference etc). This data would be invaluable in planning, scheduling and executing scientific workloads. Indeed, fundamental changes can be made to the Condor architecture itself. For example, a regular database join can be adapted to perform the task of *matchmaking* – pairing jobs and machines based on their requirements. Initial work in this area has produced encouraging results [27]. The daemons themselves can be simplified if the database is used smartly. For example, job submission can be reduced to inserting a few records in a 'Jobs' table in the database. Log files can eventually be eliminated. This architecture paves the way for a more active role for the database in tasks such as planning, job scheduling, data transfer and cluster management.

8 Challenges and conclusion

Several challenges lie in the way of a DB-integrated workflow management system. Firstly, there is an obvious 'impedance mismatch' between the execution environment offered by a database and that offered by a traditional OS. Most scientists are used to programs that handle data in a traditional file system. Storing data in relations is quite different. Databases usually require a fixed schema and do not allow arbitrarily complex objects. While traditional user programs can access files in any pattern, access to relations is usually done via cursors. A practical approach would involve allowing the database to execute external user programs and providing a real-time relational 'view' of user data, queryable via SQL. A preliminary implementation of this functionality has been made in the PostgreSQL open source ORDBMS. It allows users to specify a schema for data resident in the file system hierarchy and query this data using SQL. It supports the creation of indices on these relational views of file data. However, updates to these views via the database are not allowed. Handling complex data types that scientists are used to and synchronizing updates between the operating system and the database is still an open problem. Ideally, the DBMS must allow the integration of 'external' services such as filesystem access with traditional SQL access. This would make it possible to rewrite the more data-intensive portions of a workflow in SQL for better performance.

Becla and Wang [13] present an interesting perspec-

tive on using database techniques for scientific workloads. The authors recognized the importance of concurrency control, scalability, durability and administrative ease in dealing with two terabytes of data produced by 2000 nodes per day in the BaBar high energy physics experiment. An initial OODBMS (Objectivity/DB) based approach was eventually abandoned in favor of a hybrid RDBMS/file-based solution. The authors felt that general purpose solutions are not applicable in dealing with systems of this magnitude, and that some customization is required to tune off-the-shelf data and workflow management products to handle specific scientific computing environments.

Scientific workflows have traditionally dealt with compute-intensive tasks, while most databases are designed to minimize disk I/O. In recent years, research has been done in the optimization of queries with User-Defined Functions [18]. Like scientific tasks, UDFs are usually treated as black-boxes and are often compute-intensive. However, unlike scientific programs, UDFs are used in conjunction with SQL queries and run for seconds instead of hours. Gathering statistics such as job execution time and data access patterns is essential to planning scientific computation. It remains to be seen how such information can be obtained by the database from the execution of scientific programs. Perhaps some combination of user-provided hints and historical information will be necessary.

In conclusion, though the area of scientific data management is still young, we feel that any direction it takes will certainly incorporate database technology in a crucial way. Research in the field must reflect this. Since the contributions database research can make to the scientific realm are quite clear, it is hoped that exploring the relation between the two will open up new and exciting avenues of research for the database community.

References

[1] Biomedical informatics research network. http://www.nbirn.net.

[2] Condor dagman. http://www.cs.wisc.edu/condor/dagman/.

[3] Condor diskrouter. http://www.cs.wisc.edu/condor/diskrouter/.

[4] Condor high throughput computing. http://www.cs.wisc.edu/condor.

[5] Cyberstructure for the geosciences. http://www.geongrid.org.

[6] Grid physics network. http://www.griphyn.org.

[7] Grid physics network in atlas. http://www.usatlas.bnl.gov/computing/grid/griphyn/.

[8] Hawkeye. http://www.cs.wisc.edu/condor/hawkeye/.

[9] Ptolemy ii: Heterogenous modeling and design. http://ptolemy.eecs.berkeley.edu/ptolemyII/.

[10] Sloan digital sky survey. http://www.sdss.org.

[11] S. Abiteboul, V. Vianu, et al. Relational transducers for electronic commerce. In *PODS*, pages 179–187, 1998.

[12] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows. In *SSDBM*, pages 423–424, 2004.

[13] J. Becla and D. L. Wang. Lessons learned from managing a petabyte. In *CIDR*, pages 70–83, 2005.

[14] J. Bent, D. Thain, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and M. Livny. Explicit control in the batch-aware distributed file system. In *NSDI*, pages 365–378, 2004.

[15] A. J. Bonner. Workflow, transactions, and datalog. In *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 1999*, pages 294–305. ACM Press, 1999.

[16] M. Bote-Lorenzo and E. Dimitriadis, Y.and Gomez-Sanchez. Grid characteristics and uses: a grid definition. In *Proceedings of the First European Across Grids Conference*, pages 291–298, February 2003.

[17] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In *ICDT*, pages 316–330, 2001.

[18] S. Chaudhuri and K. Shim. Optimization of queries with user-defined predicates. *ACM Trans. Database Syst.*, 24(2):177–228, 1999.

[19] A. L. Chervenak et al. Giggle: a framework for constructing scalable replica location services. In *SC*, pages 1–17, 2002.

[20] R. L. Cole and G. Graefe. Optimization of dynamic query evaluation plans. In *SIGMOD Conference*, pages 150–160, 1994.

[21] E. Deelman, J. Blythe, et al. Pegasus: Mapping scientific workflows onto the grid. In *European Across Grids Conference*, pages 11–20, 2004.

[22] D. J. DeWitt and other. The gamma database machine project. *IEEE Trans. Knowl. Data Eng.*, 2(1):44–62, 1990.

[23] I. T. Foster, J.-S. Vöckler, M. Wilde, and Y. Zhao. Chimera: Avirtual data system for representing, querying, and automating data derivation. In *SSDBM*, pages 37–46, 2002.

[24] J. Gray et al. When database systems meet the grid. In *CIDR*, pages 154–161, 2005.

[25] K. He, S. Dong, L. Zhang, and B. Song. Building grid monitoring system based on globus toolkit: Architecture and implementation. In *CIS*, pages 353–358, 2004.

[26] Y. E. Ioannidis, M. Livny, A. Ailamaki, A. Narayanan, and A. Therber. Zoo: A desktop experiment management environment. In *SIGMOD Conference*, pages 580–583, 1997.

[27] A. Kini, S. Shankar, D. DeWitt, and J. Naughton. Matchmaking in database systems, submitted for publication. Submitted for publication.

[28] T. Kosar and M. Livny. Stork: Making data placement a first class citizen in the grid. In *ICDCS*, pages 342–349, 2004.

[29] D. T. Liu and M. J. Franklin. The design of griddb: A data-centric overlay for the scientific grid. In *VLDB*, pages 600–611, 2004.

[30] G. M. Lohman, C. Mohan, et al. Query processing in R*. In *Query Processing in Database Systems*, pages 31–47. 1985.

[31] N. W. Paton and O. Díaz. Active database systems. *ACM Comput. Surv.*, 31(1):63–103, 1999.

[32] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, pages 262–276, 2005.

An Approach for Pipelining Nested Collections in Scientific Workflows

Timothy M. McPhillips
Natural Diversity Discovery Project
tmcphillips@nddp.org

Shawn Bowers*
UC Davis Genome Center
sbowers@ucdavis.edu

ABSTRACT

We describe an approach for pipelining nested data collections in scientific workflows. Our approach logically delimits arbitrarily nested collections of data tokens using special, paired control tokens inserted into token streams, and provides workflow components with high-level operations for managing these collections. Our framework provides new capabilities for: (1) concurrent operation on collections; (2) on-the-fly customization of workflow component behavior; (3) improved handling of exceptions and faults; and (4) transparent passing of provenance and metadata within token streams. We demonstrate our approach using a workflow for inferring phylogenetic trees. We also describe future extensions to support richer typing mechanisms for facilitating sharing and reuse of workflow components between disciplines. This work represents a step towards our larger goal of exploiting collection-oriented dataflow programming as a new paradigm for scientific workflow systems, an approach we believe will significantly reduce the complexity of creating and reusing workflows and workflow components.

1. INTRODUCTION

New instrumentation, automation, computers, and networks are catalyzing high-throughput scientific research. These technologies promise to deliver data at rates orders of magnitude greater than in the past. Amid high expectations, however, is a growing awareness that existing software infrastructure for supporting data-intensive research will not meet future needs. Researchers in high-energy physics, biology, nanotechnology, climate research, and other disciplines recently reported at the 2004 Data-Management Workshops [7] that current technologies for managing large-scale scientific research do not satisfy even current needs and warned of a "coming tsunami of scientific data." They identified critical computing challenges including: (1) integrating large data sets from diverse sources; (2) capturing data provenance and other metadata; and (3) streaming data through geographically distributed experimental and computing resources in real time. An emerging challenge is the need to make high-throughput automation technologies, developed and made cost-effective by large research consortia, available to medium-sized collaborations, small research groups, and individual investigators through virtual laboratories composed of network-accessible research tools. Effective information-intensive research by small groups requires automated workflow approaches where computing infrastructure integrates disparate resources and explicitly manages project data [7].

The goals of the *Natural Diversity Discovery Project* (NDDP) [19] illuminate the challenges of supporting scientific workflows for genomics and bioinformatics. Aiming to help the public understand scientific explanations for the diversity of life, the NDDP is developing a virtual laboratory for inferring and analyzing evolutionary relationships between organisms, i.e., phylogenetic trees [8]. Professional research tools and a web-based discovery environment will enable evolutionary biologists and the general public to: (1) infer, display, and compare phylogenetic trees based on morphology, molecular sequences, and genome features; (2) correlate these phylogenies with events in Earth history using molecular clocks and the fossil record; (3) iterate over alternative phylogenetics methods, character weightings, and algorithm parameter values; (4) maintain associations between phylogenies and the data, methods, parameters, and assumptions used to infer them; (5) share workflows and results; and (6) repeat studies reported by others and note the effects of varying data sets, approaches, and parameters.

To support these research and discovery environments, the NDDP is addressing the following requirements for scientific workflows:

- **Workflows must support operations on nested collections of data.** Data sets for phylogenetics, and bioinformatics in general, can be large, complex, and nested in structure [10]. Workflows must operate efficiently on these sometimes deeply nested collections of data, while maintaining the associations they signify.

- **Workflow results must be repeatable.** Researchers need to be able to repeat the work of others easily and reliably. Workflow infrastructure must automatically record how results were obtained and allow others to use this information to reproduce the results. The provenance of input data and important intermediate results must be associated automatically with outputs.

- **Workflow definitions must be reusable.** Researchers must be able to develop generic, reusable workflows from existing workflow components. Moreover, users of the NDDP web-based discovery environment must be able to apply predefined workflows to data sets of their choice. Running a new set of data through a previously defined workflow must not entail manual reconfiguration of component parameters or interconnections.

- **Workflows and components must be robust.** Exceptions thrown for particular data or parameter sets must not disrupt operations on unrelated sets. Similarly, it should be possible for a workflow author to specify the consequences of faults during workflow processing.

*Supported in part through NSF/ITR 0225676 (SEEK).

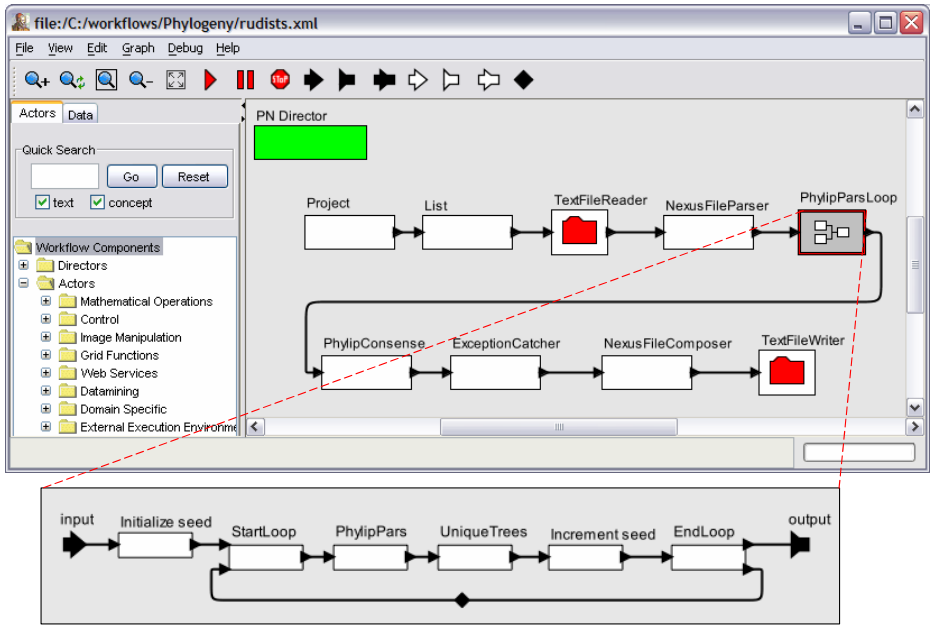


Figure 1: A collection-aware KEPLER workflow for inferring phylogenetic trees. The PhylipParsLoop *composite* actor (top) contains a nested *sub-workflow* (bottom), which iteratively executes the PARS algorithm.

We report our experiences addressing these requirements using the KEPLER scientific workflow system and detail the resulting approach. We give a brief description of KEPLER in Section 2 and describe our extensions for supporting pipelined nested data collections in scientific workflows in Section 3. Our approach treats pipelined dataflow as sequences of data tokens containing special, paired control tokens that delimit arbitrarily nested collections. We show how our implementation of these capabilities within KEPLER automates the management of nested collections and simplifies the development of "collection-aware" pipelined components. In Section 4 we describe planned extensions for supporting rich data typing of collections and workflow components. These extensions will further support the design and development of pipelined workflows and will facilitate mechanisms for workflow verification and analysis. Related work is discussed in Section 5.

2. THE KEPLER SYSTEM

KEPLER [12] is a Java-based, open-source scientific workflow system being developed jointly by a collaboration of application-oriented scientific research projects. KEPLER extends the PTOLEMY II[1] system (hereafter, PTOLEMY) with new features and components for scientific workflow design and for efficient workflow execution using distributed computational and experimental resources [16]. PTOLEMY was originally developed by the electrical engineering community as a visual dataflow programming application [14] that facilitates *actor-oriented programming* [13]. In PTOLEMY and thus in KEPLER, users develop workflows by selecting appropriate components (called *actors* or *blocks*) and placing them on the design canvas. Once on the canvas, components can be "wired" together to form the desired dataflow graph, e.g., as shown in Figure 1. Actors have *input ports* and *output ports* that provide the communication interface to other actors. Actors can be hierarchically nested, using *composite actors* to contain sub-workflows. Control-flow elements such as branches and loops are also supported (see the bottom of Figure 1). In KEPLER, actors can be written directly in Java or can wrap external components. For example, KEPLER provides mechanisms to create actors from web services, C/C++ applications, scripting languages, R[2] and Matlab, database queries, SRB[3] commands, and so on.

In PTOLEMY, dataflow streams consist of data *tokens*, which are passed from one actor to another via actor connections. PTOLEMY differs from other similar systems (including those for scientific workflows) in that the overall execution and component interaction semantics of a workflow is not determined by actors, but instead is defined by a separate component called a *director*. This separation allows actors to be reused in workflows requiring different models of computation. PTOLEMY includes directors that specify, e.g., process network, continuous time, discrete event, and finite state computation models.

As an example, the *Process Network* (PN) director executes each actor in a workflow as a separate process (or thread). Connections (or *channels*) are used to send (i.e., stream) sequences of data tokens between actors, and actors map input sequences to output sequences. Actors communicate asynchronously in process networks through buffered channels implemented as queues of effectively unbounded size. Thus, the PN director can be used to pipeline data tokens through scientific workflows, enabling highly concurrent execution.

3. PIPELINING NESTED DATA COLLECTIONS

As previously noted, scientific data sets are often large. Operating efficiently on such data sets can require pipelined processing of data set contents, a task for which scientific workflows are

[1] http://ptolemy.eecs.berkeley.edu/index.htm

[2] http://www.r-project.org/

[3] Storage Resource Broker, http://www.sdsc.edu/srb/

potentially well suited. PTOLEMY, however, does not provide explicit support for pipelining the *nested* structures typical of scientific data. PTOLEMY represents collections as individual tokens, meaning that pipelining, e.g., using process networks, occurs at the granularity of an entire collection. This approach precludes efficiency gains that might be realized by operating on collection members *concurrently* in a pipelined fashion.

To work around this limitation, workflow authors use a variety of approaches. One approach, for example, is to send the members of a collection as individual tokens, and use a separate channel (connection) to send a token count denoting the size of the collection being processed. For nested collections, this approach becomes even more complicated as it requires a large number of additional connections.

Alternatively, an actor can send special "control" tokens mixed in with the data to delimit the beginning and end of a collection (or nested collection). This approach is similar to the use of open-bracket and close-bracket information packets described by Morrison [18] and the approach proposed in Ludäscher et al [15] for supporting "pipelined arrays."[4] However, *ad hoc* use of control tokens leads to code redundancy and tightly couples actor implementation with workflow design. These problems in turn hamper rapid prototyping of workflows and associated data structures; make comprehension, reuse, and refactoring of existing workflows difficult; and limit reuse of actors designed for these workflows.

Our solution is to provide explicit support within KEPLER for pipelining nested data collections using control tokens. Specifically, we have extended KEPLER to denote nested collections using explicit, paired opening and closing delimiter tokens inserted into the data streams. We have also added to KEPLER generic operations for managing these collections.

Figure 2 illustrates how nested data collections can be streamed through consecutive actors in our approach. The collection labeled b is nested within the collection labeled a using explicit start and end control tokens. Delimited collections may contain data tokens (labeled d_i in Figure 2), explicit metadata tokens (labeled m_j in Figure 2), and other sub-collections (denoted using nested control tokens, e.g., b_{start} and b_{end}). Metadata tokens are used to carry information that applies to a collection as a whole, including data provenance. As shown in Figure 2, each actor within the pipeline is able to execute concurrently on a collection's contents. In particular, each actor is processing a part of a and its metadata simultaneously: actor 3 is processing the beginning of the a collection, actor 2 is processing the nested b collection, and actor 1 is processing the end of the a collection.

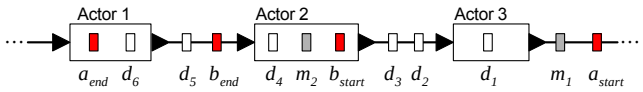


Figure 2: Pipelining data collections via explicit control tokens.

The rest of this section describes the details of our implementation. We first discuss extensions to PTOLEMY for making actors "collection-aware." We describe how an actor can easily "listen" for collections of interest. We discuss the use of collection metadata, support for provenance, and our approach to exception handling. Finally, we describe a real-world, collection-based workflow for inferring phylogenetic trees.

[4]This approach is also similar to the use of XML in stream-based query frameworks.

3.1 Collection-Aware Actors

In PTOLEMY, atomic actors are implemented by extending one of the existing actor classes. Creating a new atomic actor entails overriding the *fire* operation (as well as other related methods), which is called by a director when the actor is "activated." Within fire, an actor may read (consume) tokens from input ports, operate on input data, and write (produce) tokens to output ports. In process networks, for example, an indefinite number of tokens may be received and sent each time fire is called.

We encapsulate the complexity of creating, managing, and processing nested data collections by introducing a new type of actor called *CollectionActor*, and a new system component for managing collections called *CollectionManager*. Figure 3 shows a simplified definition of these classes. Collection-processing actors are derived from CollectionActor. Instances of CollectionManager are used to manipulate particular collections. The CollectionActor base class facilitates collection nesting by maintaining a stack of CollectionManager objects corresponding to all collections concurrently processed by an actor.

Rather than reading tokens directly from input ports, collection actors operate on collections from within methods analogous to SAX API event handlers for parsing XML files. The CollectionActor fire method triggers calls to the *handleCollectionStart* method when the opening delimiter for a collection is received; the *handleData* or *handleMetadata* method when a data or metadata token is received; and the *handleCollectionEnd* method when the closing delimiter for a collection is received. These calls pass the CollectionManager object associated with the incoming collection to these event handlers, and the newly received token to the handleData and handleMetadata methods. This event-based approach to processing collections allows collection-aware actors to be mixed freely with actors that operate on data tokens individually; the latter actors override the handleData method alone, thereby ignoring events related to collection structures and metadata.

Collection actor output is "indirect" as well. An actor may add data or metadata to a collection it is processing using methods provided by the associated CollectionManager object. An actor may create a new collection within another collection and add data or metadata to it; and it may replace data or metadata or copy the information to other collections. Collection actors specify the disposition of incoming collections, data, and metadata via the return values of the event handlers. The return value of the handleCollectionStart method declares whether the actor will further process a collection and whether the collection should be discarded or forwarded to the next actor in the workflow. Similarly, the return values of the handleData and handleMetadata methods indicate whether the token in question should be forwarded or discarded. Incoming information not discarded by an actor is streamed to succeeding actors in the workflow as the collection is received and processed.

3.2 Collection Types and Paths

Each collection is associated with a type (implemented as a Java class) denoting a (conceptual) scientific or data-management resource. For example, a collection with the type *Nexus* contains data or results associated with one or more phylogenetics computations, while a *TextFile* collection contains string tokens representing the contents of a text file.

Collection types simplify the processing of collections. Each instance of a collection actor has a *CollectionPath* parameter that specifies what conceptual types of collections and data the actor can handle. Collections and data tokens with types matching the CollectionPath value trigger collection-handling events, e.g., calls to

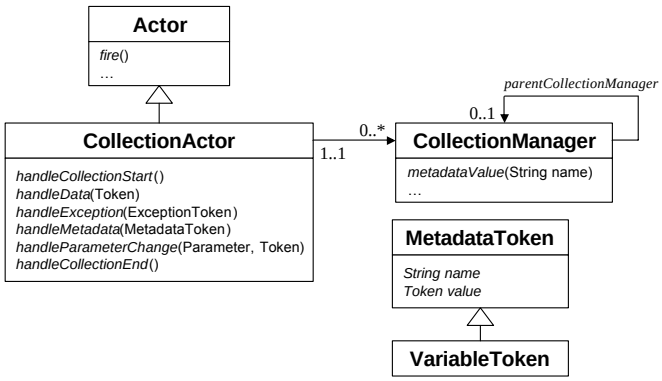


Figure 3: A simplified UML representation of the collection actor and manager classes.

the actor's handleCollectionStart and handleData methods. Collections and data not matching the CollectionPath value are streamed silently to the next actor in the workflow (i.e., no further processing is required by the actor to forward the tokens). Collection paths are expressed using a simplified XPath-style syntax expressed against type names and may specify collection types occurring anywhere in a hierarchy; parent and child collections not matching the path are ignored (although metadata for parent collections are always available). Thus, workflow developers may use the CollectionPath parameter to operate selectively on particular collection and data types, while actor authors may fix the value of these parameters to simplify actor implementation.

3.3 Context-Dependent Operations

Independent data sets passing through a workflow may require different actor behavior within a single workflow run. Actors used in such workflows must be dynamically configurable and able to operate context dependently. Metadata tokens can be used to communicate this context to actors.

An instance of the *MetadataToken* class (see Figure 3) stores the name of a metadata item and an embedded token representing the item's value. Any number of metadata tokens with distinct names may be placed within the sequence of tokens comprising a single collection.

Actors may use metadata values to tune their own behavior appropriately for the current collection. Actors may observe metadata sequentially via the *handleMetadata* method or on demand using the CollectionManager *metadataValue* method after the metadata tokens have been received. The latter random access method traverses the stack of successively enclosing collections to return the first metadata value corresponding to the given name. Thus, the context for actor behavior may be defined at any level within a set of nested collections and overridden at successively lower levels.

In KEPLER, actor *parameters* can be used to specify default actor behavior prior to workflow execution. Passing actor configuration information within collections enhances this capability. In particular, our framework allows instances of *VariableToken*, a subclass of MetadataToken, to automatically override the values of actor parameters at run time. A previous value of the parameter is restored when the end of a collection that overrides the parameter is reached. Among other advantages, variable tokens make it easy for workflow users to apply particular parameter values to subsets of data flowing through dynamically reconfigurable workflows (see the workflow described at the end of this section for an example).

3.4 Provenance Support

Collection metadata provide a convenient mechanism for recording provenance. The origin of input data, intermediate results, and workflow outputs can be described using metadata tokens, and this provenance information can be used to reproduce workflow results later. Inserting metadata directly into the data stream is an effective alternative to storing provenance information within databases or other persistent storage during workflow execution. The approach ensures that metadata is associated with the appropriate collections of data even when independent data sets are processed concurrently by pipelined workflows. It also reduces the burden on the authors of actors that do not require access to metadata: the event-based model for handling data and metadata tokens makes provenance annotations effectively "invisible ink" to actors that do not override the handleMetadata method of CollectionActor. Finally, the in-stream approach to recording provenance is convenient for distributed computing environments where all nodes may not have access to a single shared resource for storing metadata.

3.5 Exception Handling

Exception handling is a significant hurdle to supporting complex scientific workflows [16]. Exceptions can occur for many reasons. For example, many NDDP workflows include actors that wrap existing software. These legacy scientific application programs vary widely in robustness. Inappropriate input data or parameters can result in program faults. Moreover, many scientific applications are prone to crashes even when given valid instructions and data. Without mechanisms for handling exceptions in pipelined workflows, an error caused by a single data set (or collection) can result in the sudden termination of an entire workflow run.

We have addressed these issues by adding support in KEPLER for associating exceptions with collections. A collection-aware actor that catches an external application error (or other exception) may add an ExceptionToken to the collection that caused the error. This actor may then proceed to operate on the next collection. A downstream exception-catching actor can filter out collections that contain exception tokens, and may do so at a level in the collection nesting appropriate for the particular application. This approach limits the effects of exceptions to the collections that trigger them.

3.6 Example: Inferring Phylogenetic Trees

We have used all of the above KEPLER extensions in a number of NDDP workflows. One such workflow for inferring phylogenetic trees is shown in Figure 1. The workflow is run by specifying a list of files containing input data in the Nexus file format [17]. The TextFileReader actor reads these Nexus files from disk and outputs a generic TextFile collection for each; NexusFileParser transforms these text collections into corresponding Nexus collections. PhylipParsLoop is a composite actor containing the sub-workflow shown at the bottom of Figure 1. Within this sub-workflow the PhylipPars actor executes the PARS program (as a separate system process) on each Nexus collection it receives, adding the phylogenetic trees it infers to the collection. The sub-workflow iteratively executes the PARS application using the StartLoop-EndLoop construct.

The actors labeled *Initialize seed* and *Increment seed* are instances of the SetVariable class. A SetVariable actor may add or update the value of a VariableToken using a configurable expression referring to collection metadata or variable values. The first instance of SetVariable adds a VariableToken named *jumbleSeed* to each Nexus collection, while the second increments the value of this variable. The VariableToken overrides the value of the jumbleSeed parameter of the PhylipPars actor, causing the PARS ap-

plication to jumble the order of analyzed taxa differently on each execution.

The UniqueTrees actor removes redundant trees from the Nexus collection on each pass through the loop. Both PhylipPars and UniqueTrees update the *treeCount* metadata item. The loop is executed until a minimum number of unique trees has been found, or the maximum allowed number of cycles has been performed. The PhylipConsense actor applies the CONSENSE external application to the trees inferred by the PhylipParsLoop sub-workflow, adding a consensus tree (reflecting commonalities in the trees inferred by PARS) to each Nexus collection. The rest of the workflow discards Nexus collections that triggered exceptions and writes out the remaining Nexus collections to disk. The PARS and CONSENSE programs are part of the Phylip phylogeny inference package[5].

The visual simplicity of the workflow (as shown in Figure 1) highlights the power of the pipelined collection approach in KEPLER. Fairly complex, nested data collections (including trees and character matrices) are streamed through the pipeline; the actors PhylipPars and PhylipConsense wrap external scientific applications; and conditional control-flow constructs are used. All this is achieved without introducing numerous connections between actors or customizing actor Java code for the particular workflow. Moreover, the flow of data is pipelined automatically, with workflow components operating concurrently.

An *ad hoc* approach to processing collections would significantly complicate the visual appearance and development of this workflow. Even the relatively straightforward UniqueTrees actor likely would need at least two input ports, one for receiving tokens representing the trees to compare, and another for receiving a token count indicating how many trees the actor should expect to receive. This actor might require two output ports for similar reasons. Other actors in the workflow would require multiple ports and connections as well. Supporting the looping, exception handling, and metadata processing features of this workflow would introduce even more complexity. Passing aggregate tokens (e.g., tokens containing arrays of other tokens) between actors would reduce this complexity somewhat at the cost of limiting concurrent actor execution. In contrast, our delimited-collections approach not only allows multiple, independent collections to stream through workflows at the same time (e.g., if multiple Nexus input files are specified by the List actor in Figure 1), it also allows the *members* of each collection to be processed concurrently in pipelined fashion as appropriate (as illustrated in Figure 2). Most significant, the above alternative implementations would require the scientist developing the workflow to pay considerable attention to the detailed control-flow aspects of the workflow, rather than focusing primarily on the scientific tasks modeled succinctly in Figure 1.

4. TYPING EXTENSIONS

Type safety is a key element of robust programming environments. PTOLEMY provides type safety by enabling actor authors to specify the types of tokens that may pass through actor input and output ports. Exceptions are thrown if incompatible tokens are received or sent by such ports. Further, it is possible to determine prior to workflow execution if connections are type-safe. Such static workflow analysis is desirable both for workflow design and for notifying users of potential problems prior to workflow execution.

In the context of pipelining nested data collections, however, the static typing approach employed by PTOLEMY is no longer appropriate. Any type of token can occur within a stream, including metadata and delimiter token types. Only certain types of collections and data (specified, e.g., using collection paths) are processed by an actor, while all non-relevant data is forwarded transparently to downstream components. Restricting an input port to a particular structural type would unnecessarily limit the applicability of an actor to streams containing information only of that type; and restricting an output port to a particular type would cause unnecessary exceptions for all non-relevant data.

PTOLEMY type checking is disabled in our current implementation, but we intend as future work to "resurrect" the typing of actors as follows. Using a collection-based typing language, similar to content model definitions in XML Schema and DTDs (and subtyping rules, e.g., [11]), we allow actors to explicitly define the types of collections they process both conceptually (see below) and structurally. In this way, collection paths are extended to support structural collection types, expressed as restrictions on the allowable types of data tokens and sub-collections they may contain.

In addition, actor authors can provide output types specifying the types of collections (both conceptually and structurally) an actor can produce upon firing. In a static analogy to the current dynamic use of return types in the *CollectionActor* class, we allow actor authors to specify explicitly whether an input collection of a certain type will be forwarded or discarded by the actor. Actor authors can also specify whether an output collection will result from creating a new collection or by altering particular input collections. With these specifications, KEPLER can perform static type analyses, e.g., to determine that a particular connection will result in an actor never firing, or that an actor may receive a conceptually appropriate collection from an upstream actor, but with an unsupported structural type. For output types, KEPLER can also perform runtime type checking using these specifications.

The current *CollectionPath* implementation defines conceptual types using a hierarchy of Java classes. This approach works when the number of collection types is small, the types are fairly static, and the actors operating on these types are developed by the same organization. As other organizations develop their own collection-based actors, developers independently define their own collection types, and users begin mixing collection actors and types originally developed for different disciplines, more robust and richer typing mechanisms will be needed. As future work we intend to adopt the ontology-based *hybrid typing* approach [2, 3] in KEPLER to specify both structural and conceptual types of collections.

In particular, KEPLER extends the type system of PTOLEMY by separating the concerns of conventional data modeling (structural data types) from conceptual data modeling (semantic data types). A semantic type represents an ontology concept, expressed in description logic (e.g., in OWL-DL).[6] Optionally, hybrid types permit structural types and semantic types to be linked through *semantic annotations*, expressed as logical constraints. Within KEPLER, hybrid types enable concept-based searching for actors, can be used to further propagate and refine known (structural or semantic) types in scientific workflows, and can help to infer (partial) structural mappings between structurally incompatible workflow components.

5. RELATED WORK

A number of scientific workflow systems have recently emerged [1, 6, 20, 22, 21, 4]. To the best of our knowledge, none offer approaches for pipelining and managing nested data collections for workflows consisting of external and opaque (as opposed

[5]http://evolution.gs.washington.edu/phylip.html

[6]Thus, different organizations can define their own ontologies (terminologies) and easily articulate mappings among them for interoperability.

to declarative or query-based) components. We believe that KEPLER is well-suited for our extensions because of its advanced and well-defined computation models inherited from PTOLEMY, and because it provides an elegant extension mechanism via actor-oriented design. The way collection actors and managers operate on pipelined nested collections has similarities with some XML stream processing techniques [9], which thus are good candidates for collection processing optimization strategies. Finally, our approach to pipelining workflows is similar in spirit to list processing constructs in functional programming [5] as well as dataflow programming [18]. We believe that many constructs in these approaches may help make scientific-workflow design and development simpler and more intuitive for scientists.

6. CONCLUDING REMARKS

Our approach and associated KEPLER extensions facilitate the concurrent execution of collection-oriented scientific workflows. In particular, our framework automates the pipelining of individual data tokens within nested collections passing through a workflow, and at the same time explicitly maintains the inherent structure of the collections. The approach also allows metadata to be inserted into token streams on the fly; this metadata can be used for recording provenance and dynamically customizing actor behavior. Similarly, the repercussions of external application faults and other exceptions can be limited and controlled through special exception tokens. Our event-based model for handling nested collections makes collection-aware actors simple to implement and maintain, and workflows based on them easy to prototype and extend. Our approach simplifies collection-oriented scientific workflows by eliminating the need for explicit control ports and workflow-specific actors. Future extensions to support true semantic collection types will enable static analysis of workflows and will provide better support for cross-discipline and inter-organization sharing and reuse of workflow components.

The source code for the implementation described here is available in the latest release of KEPLER, which can be downloaded from the KEPLER project web site [12].

7. REFERENCES

[1] A. Ailamaki, Y. Ioannidis, and M. Livny. Scientific workflow management by database management. In *SSDBM*, 1998.

[2] C. Berkley, S. Bowers, M. Jones, B. Ludäscher, M. Schildhauer, and J. Tao. Incorporating semantics in scientific workflow authoring. In *SSDBM*, 2005.

[3] S. Bowers and B. Ludäscher. Actor-oriented design of scientific workflows. In *Proc. of the Intl. Conf. on Conceptual Modeling (ER)*, 2005.

[4] L. Bright and D. Maier. Deriving and managing data products in an environmental observation and forecasting system. In *Conf. on Innovative Data Systems Research (CIDR)*, 2005.

[5] P. Buneman, S. A. Naqvi, V. Tannen, and L. Wong. Principles of programming with complex objects and collection types. *Theoretical Computer Science*, 149(1), 1995.

[6] D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor, and I. Wang. Programming scientific and distributed workflow with Triana services. *Concurrency and Computation: Practice and Experience, Special Issue on Scientific Workflows*, 2005.

[7] The office of science data-management challenge. Report from the DOE Office of Science Data-Management Workshops, March–May 2004.

[8] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Inc., 2004.

[9] L. Golab and M. T. Özsu. Issues in data stream management. *ACM SIGMOD Record*, 2003.

[10] P. Gordon. XML for molecular biology. http://www.visualgenomics.ca/gordonp/xml/.

[11] H. Hosoya and B. C. Pierce. Regular expression pattern matching for XML. *Journal of Functional Programming*, 13(6), 2003.

[12] The Kepler Project. http://www.kepler-project.org.

[13] E. A. Lee and S. Neuendorffer. Actor-oriented models for codesign: Balancing re-use and performance. In *Formal Methods and Models for Systems*. Kluwer, 2004.

[14] E. A. Lee and T. M. Parks. Dataflow process networks. *Proc. of the IEEE*, 83(5):773–801, 1995.

[15] B. Ludäscher and I. Altintas. On providing declarative design and programming constructs for scientific workflows based on process networks. Technical report, SciDAC-SPA-TN-2003-01, 2003.

[16] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows*, 2005.

[17] D. Maddison, D. Swofford, and W. Maddison. NEXUS: An extensible file format for systematic information. *Systematic Biology*, 46(4), 1997.

[18] J. Morrison. *Flow-Based Programming*. Van Nostrand Reinhold, 1994.

[19] Natural Diversity Discovery Project. http://www.nddp.org.

[20] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. Pocock, A. Wipat, and P. Li. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics Journal*, 20(17), 2004.

[21] SciTegic. http://www.scitegic.com/.

[22] D. Weinstein, S. Parker, J. Simpson, K. Zimmerman, and G. Jones. *Visualization Handbook*, chapter Visualization in the SCIRun Problem Solving Environment. Elsevier, 2005.

WOODSS and the Web: annotating and reusing scientific workflows

C. B. Medeiros, J. Perez-Alcazar, L. Digiampietri, G. Z. Pastorello Jr, A. Santanche
R. S. Torres, E. Madeira and E. Bacarin

Institute of Computing – University of Campinas – UNICAMP CP 6176, 13084-971 Campinas, SP, Brazil

{*cmbm,jperez,luciano,gilberto,santanch,rtorres,edmundo,bacarin*}*@ic.unicamp.br*

Abstract

This paper discusses ongoing research on scientific workflows at the Institute of Computing, University of Campinas (IC - UNICAMP) Brazil. Our projects with bio-scientists have led us to develop a scientific workflow infrastructure named WOODSS. This framework has two main objectives in mind: to help scientists to specify and annotate their models and experiments; and to document collaborative efforts in scientific activities. In both contexts, workflows are annotated and stored in a database. This "annotated scientific workflow" database is treated as a repository of (sometimes incomplete) approaches to solving scientific problems. Thus, it serves two purposes: allows comparison of distinct solutions to a problem, and their designs; and provides reusable and executable building blocks to construct new scientific workflows, to meet specific needs. Annotations, moreover, allow further insight into methodology, success rates, underlying hypotheses and other issues in experimental activities.

The many research challenges faced by us at the moment include: the extension of this framework to the Web, following Semantic Web standards; providing means of discovering workflow components on the Web for reuse; and taking advantage of planning in Artificial Intelligence to support composition mechanisms. This paper describes our efforts in these directions, tested over two domains – agro-environmental planning and bioinformatics.

1 Introduction

Scientific activities involve complex multidisciplinary processes and demand cooperative work from people in distinct institutions. While the Web provides a good environment for this kind of work, on the other hand it introduces the problems of data, process and tool proliferation. Challenges in this scenario include how to understand and organize these resources and provide interoperability among tools to achieve a given goal, as well as appropriate mechanisms to document, share, deploy, construct and re-execute scientific experiments.

Scientific workflows [23] are being used to help solve some of these questions. In particular, their execution must allow for issues such as human curator intervention, flexibility in specification to accommodate distinct approaches to a problem, deviation from the specification while the workflow is being executed and unfinished executions.

Our work with environmental and bioinformatics applications have shown us that scientific workflows are moreover a good *documentation* paradigm. In other words, workflows not only help specification and execution, but keeping track of evolution of experiments and their annotation. We are testing these concepts using WOODSS (WOrkflOw-based spatial Decision Support System) [7, 11, 15, 19], a scientific workflow infrastructure developed at IC - UNICAMP. Originally conceived for decision support in environmental planning, it has evolved to an extensible environment that supports specification, reuse and annotation of scientific workflows and their components.

We are now extending this work to the Web, supporting flexibility in workflow design, experiment annotation, customization and reuse. This paper describes this work, being centered on the following issues: the database of workflow building blocks, that induces a workflow design methodology; the use of component technology to encapsulate these blocks, enabling their reuse and composition; and the exploration of advances in planning in AI to support automatic and semi-automatic composition of workflows. We briefly mention our testbed efforts in two domains - agro-environmental planning and bioinformatics.

2 Overview of WOODSS

WOODSS was initially conceived to support environmental planning activities. It was implemented on top of a commercial Geographic Information System (GIS) and has been tested in several contexts, mostly within agro-environmental applications.

The original idea was to dynamically capture user interactions with a GIS in real time, and document them by means of scientific workflows, which could then be re-executed invoking the GIS functions.

In environmental applications, user interactions

with a GIS express models for solving a given problem. Fig. 1, adapted from [11] illustrates the interaction modes of WOODSS. In the first mode, users interact with the GIS, to implement some model, whose execution is materialized into a map. WOODSS processes this interaction and generates the corresponding scientific workflow specification, stored in the Workflow Repository. In the second interaction mode, users access WOODSS' visual workflow editing and annotation interface to query, update and/or combine repository elements, annotating and constructing workflows that can be stored in the repository and subsequently executed invoking the GIS. This allows workflow evolution and lets scientists find out about previous solutions to a similar problem. The geographic database contains domain-specific data.

The present version of WOODSS is implemented in Java^{TM}, with the repository in POSTGRESQL. It contains scientific workflows (and parts thereof) and associated annotations (keywords, free text, metadata and domain ontologies [7]). The graphical interface allows user-friendly workflow editing and manipulation. More details appear in [11, 19].

Our extensions to this framework, described in the next sections, can be sketched in terms of Fig. 1. First, the geographic database is replaced by sets of databases for scientific application domains – e.g., genomics, agriculture. Second, instead of the functions of a proprietary GIS, we consider invoking third party applications/services (the "execute workflow" arrow). The "capture interaction" arrow can be replaced, for specific applications, by monitoring modules that generate the appropriate workflow specifications. Third, the Workflow Repository contains annotated (sub)workflows and their building blocks, designed according to a specific model (see Sect. 3) and encapsulated into our reuse unit Digital Content Components – DCC, see Sect. 4.

3 WOODSS' data model

This section gives an overview of the base data model used for representing scientific workflows in WOODSS; more details are available in [14, 15]. The workflow terms used here have the usual meaning, e.g., (i) a workflow is a model of a process; (ii) a process is a set of inter-dependent steps needed to complete a certain action; (iii) an activity can be a basic unit of work or another workflow; (iv) a transition establishes a dependency relation between two activities; (v) an agent is responsible for the execution of a basic unit of work; (vi) a role specifies the expected behavior of an agent executing an activity; and so on.

The model supports the storage of several abstraction levels of a workflow in a relational database. This induces a methodology for correctly constructing workflows, whereby users must first specify types of workflow building blocks, next combine them into abstract specifications (*abstract workflows*), and finally instantiate these specifications into executable (sub)workflows (*concrete workflows*).

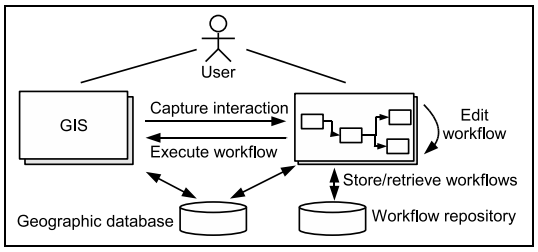


Figure 1: WOODSS interactions - adapted from [11]

The building blocks can be split in three major levels, which can be refined into many intermediate levels. The three levels reflect the methodology – see Fig. 2. The first level corresponds to definitions of data and activity types, and roles to be fulfilled by agents. Fig. 2(a) shows two activity types (*Combine Maps* and *Classify*), typical of environmental tasks. Activity types are specified in terms of their interfaces, which are based on the previously defined data types. In the figure, *Combine Maps*'s interface has two inputs, *I1* and *I2*, and one output *O1*.

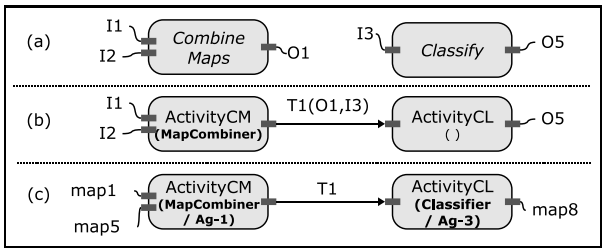


Figure 2: Levels of workflow specification

Activity and data types are used to create blocks at the second abstraction level, where a workflow structure – the abstract workflow – is specified, through transitions and dependencies among activities – e.g. the activity labeled *ActivityCM* is of type *Combine Maps* – see Fig. 2(b). At this level, it is also possible to refine activity types by associating roles to them – e.g., *ActivityCM* is associated with role *MapCombiner*. The transition *T1* connects interface elements *O1* and *I3*. The specification of types and of abstract workflows capture the notion of *workflow design* independent of execution aspects (agents and data connections), allowing design reuse, as will be seen in Sect. 4.

The last level involves creating an executable version of an abstract workflow – i.e., the concrete workflow. This is achieved by associating agents with activities and actual data sources with activities' interfaces. Fig. 2(c) shows *map1* and *map5* data sources as the input parameters for the *ActivityCM* activity, which can be executed by invoking the specific GIS *Ag-1*.

We point out four important issues covered in our model. First, the distinction among the levels is not so clearcut, since a characteristic of scientific workflows is their incremental construction – see Sect. 4. Second, the specification of an activity separating its interface from its functionality is the basis of the mechanism for allowing definition of arbitrary nesting of sub-workflows. Indeed, a sub-workflow is just another activity, accessed via its interface, and whose specification is encapsulated. A workflow/activity thus references any other workflow/activity specification inside the database, without violating the encapsulation. Third, at any moment scientists can attach annotations to the building blocks. Finally, blocks are encapsulated into DCCs – see Sect. 4.

Workflow blocks are serializable in WS-BPEL, in order to make them available on the Web. The selection of an XML-based language to represent workflows took into consideration current standard proposals. In spite of shortcomings for representing workflows, WS-BPEL turned out to be the most suitable to our representation needs [14]. More details can be found in [14, 15].

4 Reuse

Workflow building blocks – types, abstract and concrete workflows – are stored in a database for documentation and reuse. WOODSS considers two kinds of workflow reuse: that of reusing a workflow design, and that of reusing executable workflow elements. To facilitate their discovery and combination, we encapsulate these reusable units into *Digital Content Components (DCCs)* [17], special content managing units that we have been using to build applications.

DCCs are self-contained stored entities that may comprise any digital content, such as pieces of software, multimedia or text. Their specification takes advantage of Semantic Web standards and ontologies, both of which are used in their discovery process. In our workflow repository, we differentiate between two kinds of DCC – *design* and *executable* DCCs. A design DCC contains an abstract process specification, represented as a WOODSS abstract workflow (see Sect. 3). An executable DCC is a component ready to be used in some execution and encapsulates any kind of executable process description, as

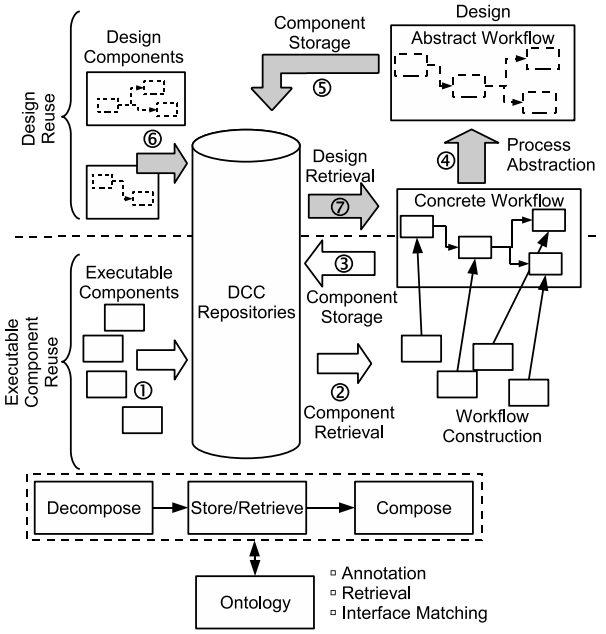


Figure 3: Reuse levels.

the concrete workflows (see Sect. 3).

DCC interfaces have been designed to semantically describe how DCCs can be accessed and combined. Interface description uses WSDL (syntactic specification) and OWL-S (`www.daml.org/services`) (semantic specification via ontologies). DCCs and Web services are homogeneously treated in our model, since they have WSDL interfaces and are able to handle the same protocols. Semantically enriched descriptions have a key role as they [18]: (i) enable similarity-based search that explores ontological relationships; (ii) support an unambiguous and more precise verification and assistance during workflow building.

Fig. 3 shows the two kinds of workflow reuse supported by WOODSS via encapsulation of building blocks into DCCs – executable components and design reuse.

Executable components reuse, represented by the cycle ①–②–③, is based on executable DCCs. This kind of DCC encapsulates third party software ① or a concrete workflow ③. Workflows are produced or modified by retrieving and composing previously stored executable DCCs ②.

In *design reuse*, represented through the cycle ④–(⑤/⑥)–⑦ we are interested in the abstract workflow inside the design DCC. Design DCCs can result from a process abstraction ④, stored in the repository ⑤; or can be produced by third party designer ⑥. Furthermore, a design DCC can be used in the

construction of a concrete workflow ⑦.

Scientific workflows can, by nature, be designed and modified on the fly during their execution. Thus, it is important to point out that the distinction, in the figure, of separate reuse cycles for design and executable DCCs has only didactical purposes, since we can intersperse both. Indeed, an abstract workflow, inside a design DCC, can have portions that are concrete specifications. Nonetheless, it still maintains its characteristics of design DCC, and may be executed considering only the concrete portions. During its execution, scientists can convert the abstract portions into concrete specifications, filling in the missing details. Other possible updates are: modification of the workflow structure (e.g., by changing control structures, deleting or including DCCs); and annotations.

Reuse via DCC is comparable to software engineering's software component reuse, with two advantages: (i) reuse is not limited to program code, but also to design encapsulated inside components; (ii) interfaces provide semantically enriched descriptions. Moreover, we also allow encapsulation of complex data within DCCs, providing thus an unified model for design, process and data. Data DCC are outside the scope of this paper – see [17].

Finally, we point out that workflow building blocks have two kinds of annotation mechanisms. First, users can associate information to them while constructing or executing a workflow (see Sect. 3) – e.g., providing pointers to documentation such as bibliography, reasons for adopting a model, and even commenting on success rate of a given experiment. Such annotations can be attached to (sub)workflows, activities, agents, data, data types, etc. Second, DCC interfaces provide semantic annotations that are used for searching and composing the appropriate workflow blocks (using among others ontology alignment solutions) [18].

5 Case studies

5.1 Agro-Environmental Management

WOODSS' workflow base started from environmental studies, and evolved to agro-environmental planning with help of experts from the Brazilian Agriculture Ministry. Agro-environmental management combines environmental (preservation) and agricultural (exploitation) issues, thus presenting an interesting challenge to scientific workflow specification. Examples of factors involved include regional topography, soil properties, climate, crop requirements, social and environmental issues. Data sources include sensors such as weather stations and satellites, and may be stored in a variety of databases, with different spatial, temporal and thematic scopes [8].

Examples of an abstract workflow specification is, for instance, a sub-workflow that computes an estimate of soil erosion for an area given a crop's needs. This would use, among others, sub-workflows such as those indicated in Fig. 2(b). This can become a concrete sub-workflow if actual data instances are provided, and agents are defined (e.g., a Web service). This sub-workflow can be composed with another one, that for instance evaluates which parts within the area have to be left alone in order to decrease erosion risk.

Within the agricultural domain, we are now investigating the interesting problem of integration of scientific workflows into agricultural supply chains [2], thereby breaching the gap between business and scientific workflows. A supply chain is a network of heterogeneous and distributed elements – retailers, distributors, transporters, storage facilities and suppliers that participate in the sale, delivery and production of a particular product.

The workflows that run within an agricultural supply chain are mainly business workflows. They are concerned with controlling production processes, and selling and delivery of goods. Although most chain activities are business driven, scientific workflows can be needed in these environments.

For instance, precision farming, land use and climate monitoring and forecasting can be supported by scientific workflows. The result of such workflows can feed planning activities within business supply chain workflows – e.g., to forecast harvest productivity and thus distribution logistics within the supply business chain. On the other hand, business workflows can produce feedback to such scientific workflows. For example, market demand for a given produce may require expanding its production at the start of the chain, which will influence factors such as area planted, water consumption and fertilizer utilization. These factors will then affect the ecological balance of the regions involved; calculating this impact involves drawing upon scientific workflows.

5.2 Planning in Bioinformatics

The appropriate composition of blocks to construct a concrete or abstract workflow is a challenge. Artificial Intelligence research in *planning* (also called plan synthesis) concerns the (semi-)automatic creation of a sequence of actions to meet a specific goal. This research domain appeared in the sixties, heavily influenced by work on automated theorem proving and operations research. Nowadays, such techniques are employed in various domains – e.g., robotics, manufacturing processes, satellite control [9]. In particular,

a new trend is to use them to solve the problem of automatic composition of Web Services [10].

We are exploring this kind of initiative to support the semi-automatic or automatic composition of (reusable) workflow blocks for bioinformatics. Bioinformatics experiments are, most of the times, composed of several related activities, so they can be modeled as workflows [4].

Scientific workflows are being utilized in *in silico* experiments at the Laboratory for Bioinformatics (LBI) (`www.lbi.ic.unicamp.br`) at IC – UNICAMP. LBI was the first Brazilian bioinformatics laboratory, being responsible for the coordination of the assembly and annotation of the *Xylella fastidiosa* genome [6]. At present, LBI is dedicating efforts in the specification and implementation of a framework for the management of bioinformatics scientific workflows [5]. For instance, a genome assembly abstract workflow is composed by, among others, sequence filters and sequence assembly activities. This can next become a concrete workflow through instantiation of data and using tools such as *crossmatch* and *phrap*.

A large part of bioinformatics workflows are designed manually, using simple resources like script languages and invocation of Web services. However, manual composition is a hard work and susceptible to errors. Therefore, it is necessary to design tools to support the composition of bioinformatics services in an automatic or interactive (semi-automatic) way [12].

In order to support these issues, we have defined a system architecture that allows semi-automatic and automatic composition. This architecture is based in the SHOP2 [20] domain independent planner. It takes advantage of the WOODSS workflow component repository to select elements to construct abstract and concrete workflows, using DCC interface specification for service description and matching.

6 Related Work

Scientific workflows appear increasingly in the literature. One trend is concerned with showing how specific kinds of systems support application domains such as bioinformatics or geosciences. Another issue is their execution, raising questions such as transaction management or grid initiatives (e.g., [22]). Grid architectures support flexible execution options, assigning each part of a workflow for execution in a distinct node, considering factors such as availability or computational power.

Our work in extending WOODSS to the Web adds another dimension to the notion of flexibility in workflow handling. Not only does it allow distinct executable components to be stored in distributed sources on the Web, but it also allows their dynamic reuse, adaptation and configuration. Furthermore, it introduces the notion of design components, which are also stored and can be retrieved and reused to construct abstract workflow specifications.

The issue of composition presents several challenges to supporting scientific activities. We consider composing DCCs (design and executable) at two levels: manual or semi-automatic composition (using AI techniques). Planning in AI has recently considered workflow composition [10]. Interactive modeling and specification is another new trend, answering the need of customizing and tailoring workflows for a specific goal [12, 20]. In general, most efforts are oriented to business workflows. An exception is [12], a pioneer framework to support interactive composition of domain independent scientific workflows based in ontologies and planning. Our model extends this proposal by supporting interactive and automatic composition of DCCs using planning strategies.

There are several proposals for composition and serialization on the Web via languages such as XPDL (XML Process Definition Language) or WS-BPEL [13]. The latter is based on using a Web services infrastructure. This facilitates the standard utilization of several distributed heterogeneous activities and data sources. We have also adopted WS-BPEL as the standard to export the content of a workflow component, even though we had to extend the language to support specific workflow structures [14].

One of our major contributions lies in reuse. Reuse technologies can be divided in two major groups [3]. One of these groups is based on *composition technologies*, adopted in our reuse approach. Composition technologies consider two players: *components* and *composition*. Components are akin to our executable DCCs, serving as building blocks in the new product construction. Compositions (our design DCCs) connect components establishing relationships between them and defining how they will work together.

The WOODSS architecture also considers these two levels, extending them to workflow building and reuse activities. The DCC model, furthermore, allows seamless integration of mechanisms to reuse design and executable elements, whereas other approaches in the literature require distinct formalisms to handle these problems. The use of this model enables workflows to invoke not only Web services, but also any kind of facility encapsulated within a DCC.

Finally, workflow execution on the Web is not our main concern at this stage. Initiatives such as CHIMERA [1], MyGRID [21, 22] worry about execution aspects. Nonetheless, WOODSS' workflows can be executed by serializing the workflow specification into a WS-BPEL document, which can be submitted

to execution on a WS-BPEL execution engine such as the one in `www.activebpel.org`.

7 Concluding Remarks

This paper presented the ongoing research on the WOODSS scientific workflow framework. Originally conceived for supporting scientific work in environmental planning, it is now being extended to distributed web-based applications for other scientific domains. The core of this work is based on creating repositories containing workflow specifications at several abstraction levels. These specifications are encapsulated in DCCs, which provide a standard search and composition interface based on metadata and semantic annotations linked to domain ontologies.

Scientists can query repositories to reuse and compose workflow elements at the design and execution stages. This allows, for instance, comparing various versions of workflows that solve a given problem.

The main contributions are: (i) discussion of issues concerning a data model that induces a methodology for workflow design; (ii) presentation of the DCC reuse model, which integrates the two different levels of workflow production – specification and execution. We support furthermore several requirements of scientific workflows, such as on-the-fly intervention and modification. The data model and WS-BPEL export facilities have been implemented [14] and DCCs manual composition has been implemented for map management in environmental applications [16]. Ongoing work concerns porting the graphical interface to the Web, creating DCCs for bioinformatics applications and checking the suitability of SHOP2 to composition, first in bioinformatics and subsequently to other domains. Another activity is the study mentioned in Sect. 5.1 of interoperation of business and scientific workflows for agriculture.

Acknowledgments. This work was partially financed by CNPq, CAPES, FAPESP and UNIFACS, CNPq WebMaps and AgroFlow projects, and a Microsoft eScience grant.

References

[1] K. M. Anderson, R. N. Taylor, and E. J. Whitehead Jr. Chimera: hypermedia for heterogeneous software development enviroments. *ACM Transactions on Information Systems (TOIS)*, 18(3):211–245, 2000.

[2] E. Bacarin, C. B. Medeiros, and E. Madeira. A Collaborative Model for Agricultural Supply Chains. In *Proc COOPIS 2004*, volume LNCS 3290, pages 319–336, 2004.

[3] T. J. Biggerstaff and C. Richter. *Reusability framework, assessment, and directions*, pages 1–17. ACM Press, 1989.

[4] M. C. Cavalcanti, R. Targino, F. Baião, S. C. Rössle, P. M. Bisch, P. F. Pires, M. L. M. Campos, and M. Mattoso. Managing structural genomic workflows using Web services. *Data & Knowledge Engineering*, 53(1):45–74, 2005.

[5] L. A. Digiampietri, C. B. Medeiros, and J. C. Setubal. A framework based in Web services orchestration for bioinformatics workflow management. In *Proc III Brazilian Workshop in Bioinformatics*, 2004.

[6] A. J. G. Simpson et al. The genome sequence of the plant pathogen Xylella fastidiosa. *Nature*, 406(1):151–157, 2000.

[7] R. Fileto. *The POESIA Approach for Services and Data Integration On the Semantic Web*. PhD thesis, IC–UNICAMP, Campinas–SP, 2003.

[8] R. Fileto, L. Liu, C. Pu, E. Assad, and C. B. Medeiros. POESIA: An Ontological Workflow Approach for Composing Web Services in Agriculture. *VLDB Journal*, 12(4):352–367, 2003.

[9] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning, Theory and Practice*. Elsevier, 2004.

[10] *Proc. of the Workshop on Planning and Scheduling for Web and Grid Services*, June 2004. `http://www.isi.edu/ikcap/icaps04-workshop/` (as of 2005-07-11).

[11] D. Kaster, C. B. Medeiros, and H. Rocha. Supporting Modeling and Problem Solving from Precedent Experiences: The Role of Workflows and Case-Based Reasoning. *Environmental Modeling and Software*, 20:689–704, 2005.

[12] J. Kim and Y. Gil. Towards Interactive Composition of Semantic Web Services. In *Proc. of the AAAI Spring Symposium on Semantic Web Services*, march 2004.

[13] Oasis-Open. OASIS Web Services Business Process Execution Language Technical Committee. `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel` (as of 2005-07-11).

[14] G. Z. Pastorello Jr. Publication and Integration of Scientific Workflows on the Web. Master's thesis, UNICAMP, 2005. In Portuguese.

[15] G. Z. Pastorello Jr, C. B. Medeiros, S. Resende, and H. Rocha. Interoperability for GIS Document Management in Environmental Planning. *Journal on Data Semantics*, 3(LNCS 3534):100–124, 2005.

[16] A. Santanchè and C. B. Medeiros. Geographic Digital Content Components. In *Proc. of VI Brazilian Symposium on GeoInformatics*, November 2004.

[17] A. Santanchè and C. B. Medeiros. Managing Dynamic Repositories for Digital Content Components. In *EDBT 2004 Workshops*, volume LNCS 3268/2004, pages 66–77, 2004.

[18] A. Santanchè and C. B. Medeiros. Self describing components: Searching for digital artifacts on the web. In *Proc. of XX Brazilian Symp. on Databases*, October 2005.

[19] L Seffino, C. B. Medeiros, J. Rocha, and B. Yi. WOODSS - A Spatial Decision Support System based on Workflows. *Decision Support Systems*, 27(1-2):105–123, 1999.

[20] E. Sirin, B. Parsia, D. Wu, J. A. Hendler, and D. S. Nau. HTN planning for Web Service composition using SHOP2. *Journal of Web Semantics*, 1(4):377–396, 2004.

[21] R. D. Stevens, H. J. Tipney, C. J. Wroe, T. M. Oinn, M. Senger, P. W. Lord, C. A. Goble, A. Brass, and M. Tassabehji. Exploring Williams-Beuren syndrome using myGrid. *Bioinformatics*, 20(suppl_1):i303–310, 2004.

[22] The myGrid Consortium. myGrid: Middleware for in silico experiments in biology. `http://www.mygrid.org.uk/` (as of 2005-07-11).

[23] J. Wainer, M. Weske, G. Vossen, and C. B. Medeiros. Scientific Workflow Systems. In *Proc. of the NSF Workshop on Workflow and Process Automation Information Systems*, 1996.

Simplifying Construction of Complex Workflows for Non-Expert Users of the Southern California Earthquake Center Community Modeling Environment

Philip Maechling (3), Hans Chalupsky (2), Maureen Dougherty (2), Ewa Deelman (2), Yolanda Gil (2), Sridhar Gullapalli (2), Vipin Gupta (3), Carl Kesselman (3), Jihie Kim (2), Gaurang Mehta (2), Brian Mendenhall (1), Thomas Russ (2), Gurmeet Singh (2), Marc Spraragen (2), Garrick Staples (1), Karan Vahi (2)
(1) Center for High Performance Computing and Communications, USC, Los Angeles, CA 90089, USA, (2) Information Sciences Institute, USC, Marina del Rey, CA 90292, (3) Southern California Earthquake Center, USC, Los Angeles CA, 90089, USA, {Corresponding Author: maechlin@usc.edu}

Abstract:
Workflow systems often present the user with rich interfaces that express all the capabilities and complexities of the application programs and the computing environments that they support. However, non-expert users are better served with simple interfaces that abstract away system complexities and still enable them to construct and execute complex workflows. To explore this idea, we have created a set of tools and interfaces that simplify the construction of workflows. Implemented as part of the Community Modeling Environment developed by the Southern California Earthquake Center, these tools, are integrated into a comprehensive workflow system that supports both domain experts as well as non expert users.

Keywords:
Workflow, Grid, Intelligent Assistant, Seismic Hazard Analysis, Data Management, Scheduling, Pegasus, Meta-scheduler, Metadata, Semantic Web

1. Introduction:

For many regions of the globe, earthquakes represent a significant hazard to life and property. This has lead geophysicists to study the earth as a system, to gain a better understanding of the complex interactions between crustal stress, fault ruptures, wave propagations through 3D velocity structures, and ultimately, ground motions. While the holy grail of this work is earthquake prediction, a more prosaic goal is to estimate the potential for earthquake damage at specific locations on the earth and this *seismic hazard analysis* (SHA) is one of the main objectives of earthquake geophysics investigation.

Recent advances have resulted in increasingly more accurate models of the different system components: fault models, rupture dynamics, wave propagation, etc. Given this progress, it becomes obvious that attention must turn to combining these diverse elements into an integrated model of the earth. It is with this goal in mind that that the Southern California Earthquake Center (SCEC) created the Community Modeling Environment (SCEC/CME) [14]. The CME is an integrated environment in which a broad user community encompassing geoscientists, civil and structural engineers, educators, city planners, and disaster response teams can have access to powerful physics-based simulation techniques for SHA. The diversity and distribution of the user community combined with the complexity of the problem space imposes some demanding requirements on any proposed solution. These include:

- The need to deliver complex computational methods to wide range of users, from non-expert (i.e. non-geophysicists) who need hazard information to domain experts who are using the environment as part of their research program.
- The need to support multiple models, and data types. For example, the community has developed alternative earth velocity models each of which has valid uses, but produces different results.
- The distributed nature and specializations of the geophysical research community leads to distributed model development with models being developed by different organizations with differing expertise and computational resources.
- The computational requirements of earthquake models require high performance and high throughput computing techniques and the computational and data resources may be dispersed across institutions and administrative domains.

To address these requirements, the approach we take is to represent computational models as scientific workflows targeted towards execution in a distributed Grid environment. While many options exist for specification and execution of scientific workflows, addressing the SCEC community requirements dictates that existing Grid workflow solutions be combined with new user-centric interfaces in unique ways. The result has been the creation of a system for the specification and execution of scientific workflows with the following features:

- User-centered knowledge based interface that helps users express geophysical problems as high-level workflow specifications.
- Knowledge-based metadata search tools integrated with data and metadata management tools allowing users to search for data by concept rather than by metadata attribute name and value.
- Workflow refinement tools based on the Virtual Data Systems (VDS) [11] toolkit for mapping workflow specifications into executable form.
- A Grid based execution environment that enables execution of workflows against domain specific software libraries, data sources, information services and national class execution and storage resources such as the National Science Foundation's TeraGrid environment.

We note that while the CME consists of many components and offers a rich set of functionalities,

we focus on the system's workflow specification and management capabilities in this paper.

2. SCEC/CME Workflow System Overview

During development of the SCEC/CME workflow system, we have found it useful to analyze the construction and execution of workflows in three phases which are: (1) workflow specification (2) data discovery and workflow refinement, and (3) grid-based workflow execution. Fig. 1 shows the tools and data stores used in SCEC/CME workflow system and how they map to these phases.

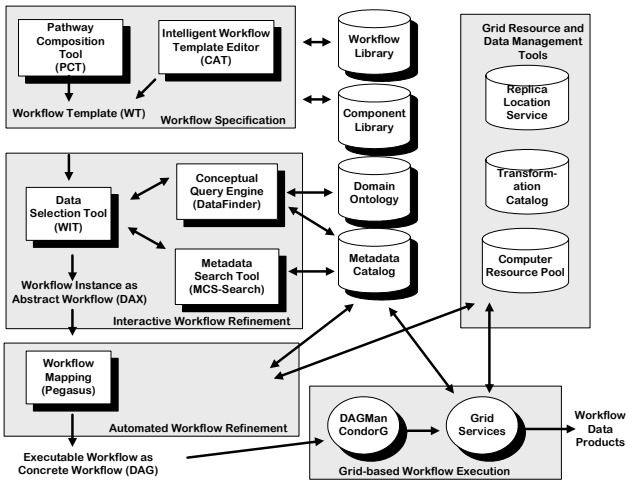


Figure 1: The SCEC/CME system supports workflow specification, refinement, and grid-based execution.

In the workflow specification phase, the user describes the problem to be solved in high-level geophysical terms and the CME assists in mapping these into high-level workflow descriptions or *workflow templates*. These descriptions define the process but do not yet include information about specific data instances or other detailed configuration information needed to execute the workflow. The data discovery and workflow refinement step fills in this missing detail, identifying data sets, selecting compute resources to execute simulation steps, determining when and where to stage data. In the final phase, a Grid-based workflow execution engine is used to execute the descriptions, keeping track of what tasks have been completed, and executing new tasks as they become ready.

One of our development challenges is to support both expert and non-expert users. A characteristic of our system is to provide the user with a selection of tools in each of these workflow phases. In order to support users with differing levels of domain and computational sophistication our system includes both manual and knowledge-based tools. For example, in the workflow specification phase, we provide a standard graphical workflow specification tool (the Pathway Composition Tool), as well as a knowledge-based intelligent assistant (Composition Analysis Tool) interface that guides users to a computational solution defined as a workflow. Also, in the data discovery and workflow refinement phase, we provide both text-matching metadata search tools as well as semantic metadata search tools to help users locate existing data.

3. Workflow Specification

The first step in generating PSHA data is to identify a simulation workflow that solves the user's problem by generating the desired data. We provide two browser-based workflow specification tools to help map problem specifications in terms of domain semantics into high-level workflow templates that define the basic computational steps in a solution.

To aid non-expert users we provide an *Intelligent Assistant* workflow construction tool called the Composition Analysis Tool (CAT) [15,16]. CAT facilitates interactive construction of computational pathways where users select and connect existing SCEC components, and the system assists in completing a correctly formulated computation. Given an outline of a workflow, CAT analyzes it using the semantic description of components, reports errors and gaps, and generates specific suggestions on how to fix these errors. Users often construct a workflow template using CAT by specifying a starting data type and a target data type and then CAT provides suggestions on which computational modules are needed in the workflow to transform the starting data type to the target data type.

The CAT system is implemented in two components; 1) a browser-based user interface tool that guides the user through the creation of a workflow template, and 2) a web service-based reasoner (called the Composition Analysis Tool Service (CAT-S)) that includes the reasoning and validation algorithms. One of the features of CAT is that it can check workflows for validity, i.e.: (a) all links are consistent with the component descriptions and their input and output data types, (b) all computational steps have linked inputs, (c) an end result has been specified, (d) all computational steps are executable, and (e) all computational steps contribute to the results. CAT uses the Web Ontology Language (OWL) standard as its knowledge representation language so the component ontology used by CAT-S for validation is expressed in OWL.

For expert users we also provide a fairly standard browser-based graphical workflow editing tool called the Pathway Composition Tool (PCT). Both tools (CAT and PCT) output workflow templates in a common XML format. However, in the case of PCT, the output workflow template has not been validated while a workflow template produced by CAT has been validated based on the semantic description of the task and constituent components.

4. Interactive Workflow Refinement

Given a specification of the processing steps that need to be performed, the workflow must be refined to identify what input data to apply the processing to, what implementations to use and where to perform the computations. The first step in this process is to identify the input data to be used for the workflow.

Before a workflow template can be used, the user must identify what data sets to which the workflow should be applied. We refer to the process of discovering, selecting, and specifying initial input files as the interactive workflow refinement phase.

In the SCEC/CME system, a user performs interactive workflow refinement using a tool we call the Workflow Instantiation Tool (WIT). WIT is a browser-based tool with which the user selects the input data instances to be used in the workflow computation. Data instances that will be calculated during the workflow execution do not need to be specified. However, data files that represent initial inputs to the workflow must be specified.

The WIT interface presents the user with each computational step in the workflow under refinement, and it shows the input data types, the computation type, and the output data types for each step. For each non-computed input data file, it presents the user with a list of existing data files of the appropriate type. WIT obtains information about the existing data files in the workflow system through the use of a Metadata Catalog Service (MCS) [7] that contains entries for all data files instances, and their associated data types, currently in the system. MCS metadata is represented by name-value pairs where each data type in the SCEC/CME system has a minimum set of required metadata attributes. As workflows execute, the workflow computational modules generate the required metadata for the data files that they produce and they register this metadata into the MCS.

In order to enable file replication, the MCS does not store file names directly but rather records a globally unique name in the form of a Uniform Resource Identifier (URI). When data files are imported into the system, or are created by computational modules, we form a URI by concatenating a SCEC namespace, a data type name, and a sequence number generated from the database. The URI can be mapped to one or more actual file names by another component of the SCEC/CME system called the Globus Replica Location Service (RLS).

If the user knows the URI of the desired input data, they may be specified directly. Otherwise, the SCEC/CME system provides two browser-based data discovery tools to help the user locate the appropriate files. One of the tools, designed for sophisticated users, provides string-based metadata attribute search capabilities. The other data discovery tool, designed to support non-expert users, provides concept-based metadata searches. With both data discovery tools, the user specifies some information describing the desired file(s) and the search tools query the system metadata to return the URIs of matching files.

The first tool, called MCS-Search, is a simple, metadata attribute-based, search tool. The user enters attribute names and values, and the system uses string-based matching to locate logical files with the desired attributes.

To improve upon this basic name-value, string-matching metadata search capability, we have also developed a semantic metadata search tool called DataFinder based on the PowerLoom knowledge representation system [18]. DataFinder utilizes a concept-based domain and metadata attribute ontology that links geophysical and seismic hazard domain concepts with the metadata attributes that describe the computational products.

DataFinder provides semantic overlays for the existing metadata attributes, enriching the information content. The DataFinder domain and metadata attribute ontology is represented in the PowerLoom representation language, based on KIF [13]. DataFinder is implemented using a hybrid reasoning approach based on combining the strengths of the PowerLoom logical reasoning engine with the database technology underlying the MCS.

The direct connection between DataFinder and the MCS database is achieved via PowerLoom's database access layer, which provides DataFinder with the necessary scalability to handle the task of locating relevant data products in a large repository. It also allows us to add semantic enhancements by overlaying the raw metadata with a hierarchy of concepts, providing for more abstract views. The DataFinder system translates the domain concepts specified by the user into SCEC/CME metadata attributes. The DataFinder search system allows users to find data instances without requiring detailed knowledge of the (multiple) specific metadata attributes used to describe the data.

When all initial, non-computed, input data files have been discovered, the user can interactively refine the workflow specification by specifying a URI for each input file. Once URIs have been specified for all non-calculated data instances, interactive workflow refinement is complete.

The output of the interactive workflow refinement phase is a representation of the workflow we call an *abstract workflow*. An abstract workflow specifies the workflow using logical references to both the files, and to the computational modules to be used. This has several advantages. By specifying the data file instances logically, the automated workflow refinement parts of the system can select the most appropriate file replicas. By specifying the transformations logically, later elements of the workflow system can select both the transformation instance and computing resources to be used during execution.

5. Automated Workflow Refinement

In the automated workflow refinement phase, we use a sophisticated execution planning system to convert an abstract workflow into a concrete workflow. In this process, URIs must be mapped to actual input files, alternative implementations for computational steps chosen and execution sites selected. In addition, we must orchestrate the movement of data between computational sites as required by the execution sites and data sources.

This conversion is performed during the automated workflow refinement phase of our system using Pegasus (Planning for Execution in Grids) [4,5,6]. The Pegasus system performs automated workflow refinement utilizing a variety of tools including the Virtual Data System (VDS) that

includes Chimera, Condor DAGMan, MCS, RLS, and Globus. Program scheduling and execution tools are based on the high throughput Condor job submission and scheduling tools.

The automated workflow refinement capabilities of the Pegasus system are one more way that the SCEC/CME system shields users from workflow complexities. Non-expert users are frequently not familiar with computing details such as how to match executables to hosts. Our system supports these users by establishing transformation to compute resource mappings in data stores such as the VDS transformation catalog and then using Pegasus to automatically select appropriate execution environments for each transformation in a workflow.

Pegasus interacts with other VDS and Globus components to drive the process of refinement of workflows from an abstract format to an executable concrete workflow specification. These include a transformation catalog that contains descriptions of the SCEC/CME transformations and RLS which is used to map URIs specified during interactive refinement into specific file instances.

During the conversion from DAX to DAG, the system queries RLS and maps each URI to an appropriate physical file name based on the location of the replicas and the location where the data file needs to be read, and replaces each URI with a URL. The system also converts each logical transformation name to physical transformation name. Pegasus also selects an appropriate computer resource on which to run the transformation by mapping each transformation's required computing capabilities to available computing resources as specified in the Pegasus resource pool configuration file. If Pegasus determines there is more than one computing resource available for use for a computational module, it selects one of the appropriate grid resources using a user selected algorithms. Supported resource selection algorithms include both round robin and random approaches.

During the conversion from abstract workflow to concrete workflow, Pegasus may augment a workflow by adding data or module staging into the workflow if the data and executable movements are required to execute on the selected computing resources. If, as Pegasus converts an abstract workflow to a concrete workflow, it recognizes that data movement is required by the workflow to run on the selected grid-resources, it will automatically add the data movement module into the workflow.

The automated workflow refinement phase in the SCEC/CME workflow system shields users from the complexities of the grid-based execution environment. Development of the resource descriptions such as the transformation catalog and the resource pool configuration can be performed by specialists knowledgeable about these environments and through Pegasus, deliver the flexible computing and data management capabilities of the Grid to non-expert users.

6. Grid-based Execution

Once the SCEC/CME system has produced a DAG, the workflow is ready for execution on a Globus-based grid. Grid-based execution is the final phase our workflow system.

As our workflow is now represented as an executable DAG, we submit the DAG to the Condor DAGMan workflow execution system. Since the Condor DAGMan interacts with Globus tools, a workflow constructed by our non-expert user now executes securely distributed across a wide variety of computing and storage devices accessible through the SCEC grid. Our grid-based workflows benefits from the job submission and scheduling capabilities of the Condor DAGMan tools. DAGMan analyzes a workflow for parallel elements and will run the computational steps in parallel where possible.

During grid-based execution, the SCEC/CME system provides workflow monitoring tools that allow users to follow the progress of their workflow as it executes on the grid. As a workflow executes and produces new data instances, the system assigns logical file names to the new files, and inserts logical file name to physical file name mappings into RLS. Modules output metadata which is written into MCS to maintain metadata for each new data instance.

At the completion of the workflow, the solution to the user's geophysical problem is stored in the system as a data file and an associated metadata description. This data instance is now available for use an input into the next geophysical problem that the scientist wishes to solve address using a workflow-based solution.

7. Case Study: Probabilistic Hazard Maps

As a discipline, explorations in geophysics are particularly well suited to be represented as scientific workflows. *Probabilistic seismic hazard analysis* (PSHA) hazard curve calculations are a good case in point. A PSHA hazard curve shows the probability that a specific site will exceed a specified amount of ground motion due to likely earthquakes over some period of time, for example, 50 years. A collection of PSHA hazard curves can be combined into probabilistic hazard maps.

The issues of component selection, configuration and computation management are complex even for members of the geophysics community. For a practicing engineer or city planner, the complexity can be overwhelming.

Representing PSHA calculation as a workflow mitigates many of these issues. Workflows help manage the complexity of the analysis in terms of the number of different types of calculations that must be performed, the number of alternatives for each processing step, and the number of data files that must be managed. Also, by enabling configuration and reuse of a pre-defined set of automatically configurable *workflow templates* we can make it possible to deliver this complex analysis to unsophisticated users, enabling them to generate PSHA maps on demand, customized to their use case requirements.

SCEC scientists utilized the SCEC/CME workflow system to perform a series of hazard map

calculations [9]. In these studies, a probabilistic seismic hazard map workflow was modeled in our system as a three transformation workflow. We will discuss this example to help illustrate the capabilities of our system.

In this example, the user wants to calculate a hazard map for the southern California region. The user begins in the workflow specification phase. To construct this workflow, the user begins the workflow specification using the CAT interface.

One way a non-sophisticated user may interact with CAT is to specify the target data type that they want to create. In this case the user wants to create a Hazard Map which is identified by the system as a *Hazard Map JPEG*. Once the user has identified this target data type, CAT is now able to suggest to the user which transformation are required to generate a files of this type. By interacting with CAT, and adding transformations into the workflow, the user develops a workflow-based solution to their problem. The workflow developed by the user with the assistance of the CAT is shown in Fig 2.

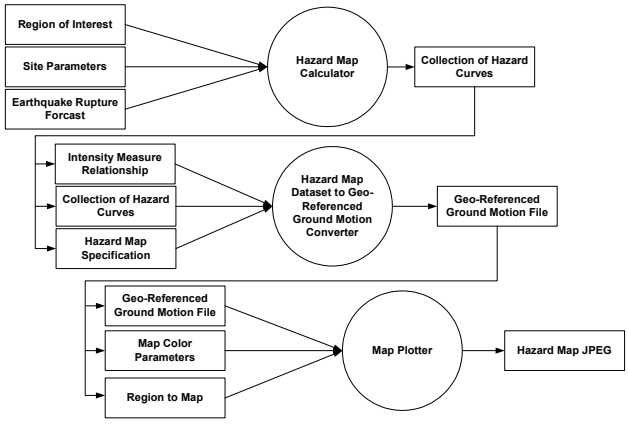


Figure 2: Probabilistic Seismic Hazard Map calculation defined as a three step workflow.

The workflow template produced by CAT indicates that each of the transformations requires two or more input data files. The figure shows that as the workflow executes output data sets will be used as inputs to subsequent transformations. However, some initial data sets must be supplied by the user in order to run this workflow. There are three user supplied initial inputs to the *Hazard Map Calculator*. The transformation called *Hazard Map Dataset to Geo-Referenced Ground Motion Converter* requires three inputs. One of these inputs will be calculated during the workflow. The other two inputs must be specified before the workflow can be run. The transformation called *Map Plotter* also requires three inputs, one of which is calculated during the workflow, and two of which must be specified prior to workflow execution.

Next, the user performs data discovery to find appropriate existing data files. For example, one input to the *Hazard Map Calculator* is a file of type *Region of Interest*. This file contains a description of the geographical boundaries of the seismic hazard study. The user can search for a file with the desired boundaries using either the MCS-Search or DataFinder tools.

The user now begins the interactive workflow refinement. The WIT tool shows the user each transformation indicating the user supplied inputs and the calculated inputs. For each user-specified input file, WIT provides the user with a list of available files of the correct type. Once the user has selected a specific logical file name for each of the seven user-supplied, initial inputs to this workflow, the user submits the abstract workflow to the Pegasus system.

The Pegasus system now begins automated workflow refinement. The abstract workflow is converted to a concrete workflow. By making calls to the RLS system, each logical file name in the DAX is converted to a physical file name. The logical transformation names are converted to actual executable names. A hazard map calculation for the southern California region may require 10,000 hazard curve calculations and our workflow system must manage this number of files for each map.

Once Pegasus has generated a DAG, the DAG is submitted to the Condor DAGMan for job submission tools for execution on the grid. The data files created during the workflow execution are entered into the MCS and RLS.

When the workflow is submitted for execution, the system returns a logical file name identifying the desired final result of our workflow, which in this case is a *Hazard Map JPEG*. This file represents the solution to our geophysical problem, and it can be accessed as soon as the workflow completes.

8. Related Work

There has been increasing effort to enable scientists to create and manage complex scientific workflows. The myGrid project [21] exploits semantic web technology to support data intensive bioinformatics experiments in a grid environment. The semantic description of services in RDF and OWL is used for service discovery and match-making. They also provide interactive tools called Taverna for authoring and executing workflows. Kepler [1] is also a data-driven workflow system where the user can author data processing steps as a network of pre-defined workflow components called ‘actors’ and use ‘directors’ for describing execution models. The system allows semantic annotations of data and actors, and can support semantic transformation of data. Triana [3] allows the user to specify execution behavior easily by supporting an abstract layer for Grid computing called GAP, a subset of the GridLab GAT. Using a graphical interface, the user can drag and drop workflow component to form a workflow and also specify distribution policy for a group of nodes in the workflow. GAP services can be advertised, discovered and communicated with using abstract high level calls on Grids and P2P networks. Our work presents complementary capabilities in that the workflow composition tools in these systems provide limited assistance in validating user authored

workflows and providing suggestions to generate complete and consistent workflows. Also these systems do not employ sophisticated resource management approaches and cannot fully make use of grid resources.

There are many Grid tools that are developed to help end-users manage complex workflows. ICENI (Imperial College e-Science Network Infrastructure) [17] provides a component based Grid middleware. Users can construct an abstract workflow from a set of workflow components and the system generates a concrete workflow using its scheduler. ICENI also support different views of composed workflows: it uses spatial view to allow flexibility during composition and provides temporal view to support scheduling optimization. GridAnt [20] developed by Argonne National Laboratory is a client-side workflow system that assists users to express and control execution sequences and test Grid services. It uses Java Cog Kit that provides access to Grid services through a Java framework. GridFlow [2] provides user portal and services of both global grid workflow management and local grid sub-workflow scheduling. Unicore [19] provides a programming environment where users can design and execute job flows with advanced flow controls. Our workflow system presents a more comprehensive grid-based environment for scientific workflows and we believe that integrating our approach with these capabilities may result in improved environments to support complex scientific workflows.

9. Discussion

The SCEC/CME system helps geoscientist execute a wide variety of geophysical simulation codes including serial Probabilistic Seismic Hazard calculations, MPI-based high performance earthquake wave propagation simulations, and I/O intensive volumetric data post-processing software.

The SCEC/CME Workflow system represents a comprehensive grid-based workflow environment that addresses a full range of workflow capabilities from interactive composition of workflows, to data and metadata management, semantic metadata search capabilities, and job scheduling that makes full use of grid resources. The system‘s use of well-established workflow tools including Globus and VDS enhances its stability, robustness, and maintainability. Through its support for grid environment, the system supports workflows that run from simple Condor Pool-based applications to high performance computing system on the TeraGrid.

The SCEC/CME system utilizes standard semantic web technologies including OWL, SOAP, and WSDL to enhance platform independence and interoperability. The system supports a wide variety of computational modules, including, but not limited to web service invocations. This support for a variety of computational module types eases the integration of existing programs into the workflow system.

During this work, we simplified user interactions with the system by introducing intelligent assistants and knowledge driven resource selections. These tools utilize a variety of data stores including computing and domain ontologies, as well as metadata, computational module, and computer resource descriptions. It is through these descriptive, rather than computational, data stores, that we are able to provide user assistance, and to simplify the user experience.

In our view, the development and use of these knowledge repositories shifts the burden of detailed knowledge of workflows, in an appropriate way, towards our domain and computing experts and away from end users of our workflow system. By capturing the knowledge of domain and computing experts in knowledge bases, and by constructing our tools to use this captured knowledge, we reduced the expertise required to use our scientific workflow system making the system more accessible to non-expert users.

Acknowledgements

This work was support by the SCEC Community Modeling Environment Project which is funded by the National Science Foundation (NSF) under contract EAR-0122464 (The SCEC Community Modeling Environment (SCEC/CME): An Information Infrastructure for System-Level Earthquake Research). This work was also supported by the NSF GriPhyN Project, grant ITR-800864. SCEC is funded by NSF Cooperative Agreement EAR-0106924 and USGS Cooperative Agreement 02HQAG0008. The SCEC contribution number for this paper is 915.

10. References

[1] Altintas, I., C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, S. Mock, Kepler: Towards a Grid-Enabled System for Scientific Workflows, In the Workflow in Grid Systems Workshop in GGF10 - The Tenth Global Grid Forum, Berlin, Germany, March 2004.

[2] Cao, J., S. A. Jarvis, S. Saini, and G. R. Nudd. GridFlow: Workflow Management for Grid Computing. In 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003), Tokyo, Japan, May 2003.

[3] Churches, D., G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor and I. Wang, Programming Scientific and Distributed Workflow with Triana Services. In Grid Workflow 2004 Special Issue of Concurrency and Computation: Practice and Experience, to be published, 2005.

[4] Deelman, E., James Blythe, Yolanda Gil, Carl Kesselman, Scott Koranda, Albert Lazzarini, Gaurang Mehta, Maria Alessandra Papa, Karan Vahi (2003a). Pegasus and the Pulsar Search: From Metadata to Execution on the Grid, Applications Grid Workshop, PPAM 2003, Czestochowa, Poland 2003.

[5] Deelman E., James Blythe, Yolanda Gil, Carl Kesselman (2003b). Workflow Management in GriPhyN, Grid Resource Management, Kluwer, 2003.

[6] Deelman, E., James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, Kent Blackburn, Albert Lazzarini, Adam Arbree, Richard Cavanaugh, and Scott Koranda (2003c). Mapping

Abstract Complex Workflows onto Grid Environments, Journal of Grid Computing, Vol.1, no. 1, 2003, pp. 25-39.
[7] Deelman, E., James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, Miron Livny (2004), Pegasus : Mapping Scientific Workflows onto the Grid, Across Grids Conference 2004, Nicosia, Cyprus, 2004
[8] Domenico B.,John Caron, Ethan Davis, Robb Kambic and Stefano Nativi (2002). Thematic Real-time Environmental Distributed Data Services (THREDDS): Incorporating Interactive Analysis Tools into NSDL, Journal of Digital Information, Volume 2 Issue 4 Article No. 114, 2002-05-29
[9] Field, E.H.,V. Gupta, N. Gupta, P. Maechling, and T.H Jordan (2005). Hazard Map Calculations Using GRID Computing**,** to be published to *Seismological Research Letters*, 2005
[10] Foster, I., C. Kesselman, S. Tuecke,The Anatomy of the Grid: Enabling ScalableVirtual Organizations, International J. Supercomputer Applications, 15(3), 2001.
[11] Foster, I., J. Voeckler, M. Wilde, and Y. Zhao. Chimera: A Virtual Data System for Representing, Querying and Automating Data Derivation Proceedings of the 14th Conference on Scientific and Statistical Database Management, Edinburgh, Scotland, July 2002.
[12]Frey, J. T. Tannenbaum, I. Foster, M. Livny, S. Tuecke, Condor-G: A Computation Management Agent for Multi-Institutional Grids,Cluster Computing, 5(3):237-246, 2002.
[13] Genesereth, M.R. (1991). Knowledge interchange format. In J. Allen, R. Fikes, and E. Sandewall, editors, Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning, pages 599--600, San Mateo, CA, USA, April 1991.
[14] Jordan, T. H., P. J Maechling, and the SCEC/CME Collaboration (2003). The SCEC community modeling environment: an information infrastructure for system-level science, Seism. Res. Lett. 74, 324-328
[15] Kim, J., M. Spraragen, and Y. Gil (2004a), An Intelligent Assistant for Interactive Workflow Composition, In Proceedings of the International Conference on Intelligent User Interfaces (IUI-2004); Madeira, Portugal, 2004.
[16] Kim, J., M. Spraragen, and Y. Gil (2004b), A Knowledge-Based Approach to Interactive Workflow Composition, In Proceedings of the 2004 Workshop on Planning and Scheduling for Web and Grid Services, at the 14th International Conference on Automatic Planning and Scheduling (ICAPS 04); Whistler, Canada, 2004.
[17] McGough, S., L. Young, A. Afzal, S. Newhouse, and J. Darlington, WorkflowEnactment in ICENI In UK e-Science, All Hands Meeting, p. 894--900, Nottingham, UK, Sep. 2004
[18] PowerLoom: http://www.isi.edu/isd/LOOM/PowerLoom/
[19] Romberg, R., The UNICORE Architecture-Seamless Access to Distributed Resources, Proceedings of 8th IEEE International Symposium on High Performance Computing, Redondo Beach, CA, USA, August 1999, pp. 287-293
[20] von Laszewski,G. K. Amin, M. Hategan, N. J. Zaluzec, S. Hampton, and A. Rossi, GridAnt: A Client-Controllable Grid Workflow System. In 37th Annual Hawaii International Conference on System Sciences (HICSS'04) IEEE Computer Society Press, Los Alamitos, CA, USA, January 5-8, 2004.
[21] Wroe C., C.A. Goble, M. Greenwood, P. Lord, S. Miles, J. Papay, T. Payne, L. Moreau (2004), Automating Experiments Using Semantic Data on a Bioinformatics Grid, In IEEE Intelligent Systems special issue on e-Science Jan/Feb 2004.

A Survey of Data Provenance in e-Science

Yogesh L. Simmhan Beth Plale Dennis Gannon
Computer Science Department
Indiana University, Bloomington, IN 47405
{ysimmhan, plale, gannon}@cs.indiana.edu

ABSTRACT

Data management is growing in complexity as large-scale applications take advantage of the loosely coupled resources brought together by grid middleware and by abundant storage capacity. Metadata describing the data products used in and generated by these applications is essential to disambiguate the data and enable reuse. Data provenance, one kind of metadata, pertains to the derivation history of a data product starting from its original sources.

In this paper we create a taxonomy of data provenance characteristics and apply it to current research efforts in e-science, focusing primarily on scientific workflow approaches. The main aspect of our taxonomy categorizes provenance systems based on why they record provenance, what they describe, how they represent and store provenance, and ways to disseminate it. The survey culminates with an identification of open research problems in the field.

1. Introduction

The growing number and size of computational and data resources coupled with uniform access mechanisms provided by a common Grid middleware stack is allowing scientists to perform advanced scientific tasks in collaboratory environments. Scientific workflows are the means by which these tasks can be composed. The workflows can generate terabytes of data, mandating rich and descriptive metadata about the data in order to make sense of it and reuse it. One kind of metadata is provenance (also referred to as lineage and pedigree), which tracks the steps by which the data was derived and can provide significant value addition in such data intensive e-science projects.

Scientific domains use different forms of provenance and for various purposes. Publications are a common form of representing the provenance of experimental data and results. Increasingly, Digital Object Identifiers (DOIs) [1] are used to cite data used in experiments so that the papers can relate the experimental process and analysis – which form the data's lineage – to the actual data used and produced. Some scientific fields go beyond this and store lineage information in a machine accessible and understandable form. Geographic information system (GIS) standards suggest that metadata about the quality of datasets should include a description of the lineage of the data product to help the data users to decide if the dataset meets the requirement of their application [2]. Materials engineers choose materials for the design of critical components, such as for an airplane, based on their statistical analysis and it is essential to establish the pedigree of this data to prevent system failures and for audit [3]. When sharing biological and biomedical data in life sciences research, presence of its transformation record gives a context in which it can be used and also credits the author(s) of the data [4]. Knowledge of provenance is also relevant from the perspective of regulatory mechanisms to protect intellectual property [5]. With a large number of datasets appearing in the public domain, it is increasingly important to determine their veracity and quality. A detailed history of the data will allow the users to discern for themselves if the data is acceptable.

Provenance can be described in various terms depending on the domain where it is applied. Buneman et al [6], who refer to data provenance in the context of database systems, define it as the description of the origins of data and the process by which it arrived at the database. Lanter [7], who discusses derived data products in GIS, characterizes lineage as information describing materials and transformations applied to derive the data. Provenance can be associated not just with data products, but with the process(es) that enabled their creation as well. Greenwood et al [8] expand Lanter's definition and view it as metadata recording the process of experiment workflows, annotations, and notes about experiments. For the purposes of this paper, we define data provenance as information that helps determine the derivation history of a data product, starting from its original sources. We use the term data product or dataset to refer to data in any form, such as files, tables, and virtual collections. The two important features of the provenance of a data product are the ancestral data product(s) from which this data product evolved, and the process of transformation of these ancestral data product(s), potentially through workflows, that helped derive this data product.

In this survey, we compare current data provenance research in the scientific domain. Based on an extensive survey of the literature on provenance [9], we have developed a taxonomy of provenance techniques that we use to analyze five selected systems. Four of the projects use workflows to perform scientific experiments and simulations. The fifth research work investigates provenance techniques for data transformed through queries in database systems. The relationship between

workflows and database queries with respect to lineage is evident[1]. Research on tracking the lineage of database queries and on managing provenance in workflow systems share a symbiotic relationship, and the possibility of developing cross-cutting techniques is something we expose in this study. We conclude this survey with an identification of open research problems. The complete version of this survey [9] reviews an additional four systems and also investigates the use of provenance in the business domain.

While data provenance has gained increasing interest recently due to unique desiderata introduced by distributed data in Grids, few sources are available in the literature that compare across approaches. Bose et al [10] survey lineage retrieval systems, workflow systems, and collaborative environments, with the goal of proposing a meta-model for a systems architecture for lineage retrieval. Our taxonomy based on usage, subject, representation, storage, and dissemination more fully captures the unique characteristics of these provenance systems. Miles et al [11] study use cases for recording provenance in e-science experiments for the purposes of defining the technical requirements for a provenance architecture. We prescribe no particular model but instead discuss extant models for lineage management that can guide future provenance management systems.

2. Taxonomy of Provenance Techniques

Different approaches have been taken to support data provenance requirements for individual domains. In this section, we present a taxonomy of these techniques from a conceptual level with brief discussions on their pros and cons. A summary of the taxonomy is given in **Figure 1**. Each of the five main headings is discussed in turn.

2.1 Application of Provenance

Provenance systems can support a number of uses [12, 13]. Goble [14] summarizes several applications of provenance information as follows:

- *Data Quality*: Lineage can be used to estimate data quality and data reliability based on the source data and transformations [4]. It can also provide proof statements on data derivation [15].
- *Audit Trail*: Provenance can be used to trace the audit trail of data [11], determine resource usage [8], and detect errors in data generation [16].
- *Replication Recipes*: Detailed provenance information can allow repetition of data derivation, help maintain its currency [11], and be a recipe for replication [17].
- *Attribution*: Pedigree can establish the copyright and ownership of data, enable its citation [4], and determine liability in case of erroneous data.
- *Informational*: A generic use of lineage is to query based on lineage metadata for data discovery. It can also be browsed to provide a context to interpret data.

2.2 Subject of Provenance

Provenance information can be collected about different resources in the data processing system and at multiple levels of detail. The provenance techniques we surveyed focus on data, but this data lineage can either be available explicitly or deduced indirectly. In an explicit model, which we term a *data-oriented* model, lineage metadata is specifically gathered about the data product. One can delineate the provenance metadata about the data product from metadata concerning other resources. This contrasts to a *process-oriented*, or indirect, model where the deriving processes are the primary entities for which provenance is collected, and the data provenance is determined by inspecting the input and output data products of these processes [18].

The usefulness of provenance in a certain domain is linked to the *granularity* at which it is collected. The requirements range from provenance on attributes and tuples in a database [19] to provenance for collections of files, say, generated by an ensemble experiment run [20]. Increasing use of abstract datasets [17, 18] that refer to data at any granularity or format allows a flexible approach. The cost of collecting and storing provenance can be inversely proportional to its granularity.

2.3 Representation of Provenance

Different techniques can be used to represent provenance information, some of which depend on the underlying data processing system. The manner in which provenance is represented has implications on the cost of recording it and the richness of its usage. The two major approaches to representing provenance information use either annotations or inversion. In the former, metadata comprising of the derivation history of a data product is collected as *annotations* and descriptions about source data and processes. This is an *eager* form [21] of representation in that provenance is pre-computed and readily usable as metadata. Alternatively, the *inversion* method uses the property by which some derivations can be inverted to find the input data supplied to them to derive the output data. Examples include queries and user-defined functions in databases that can be inverted automatically or by explicit functions [19, 22, 23]. In this case, information about the queries and the output data may suffice to identify the source data.

While the inversion method has the advantage of being more compact than the annotation approach, the information it provides is sparse and limited to the derivation history of the data. Annotations, on the other hand, can be richer and, in addition to the derivation history, often include the parameters passed to the derivation processes, the versions of the workflows that

[1] Workflows form a graph of processes that transform data products. Database queries can form a graph of operations that operate on tables.

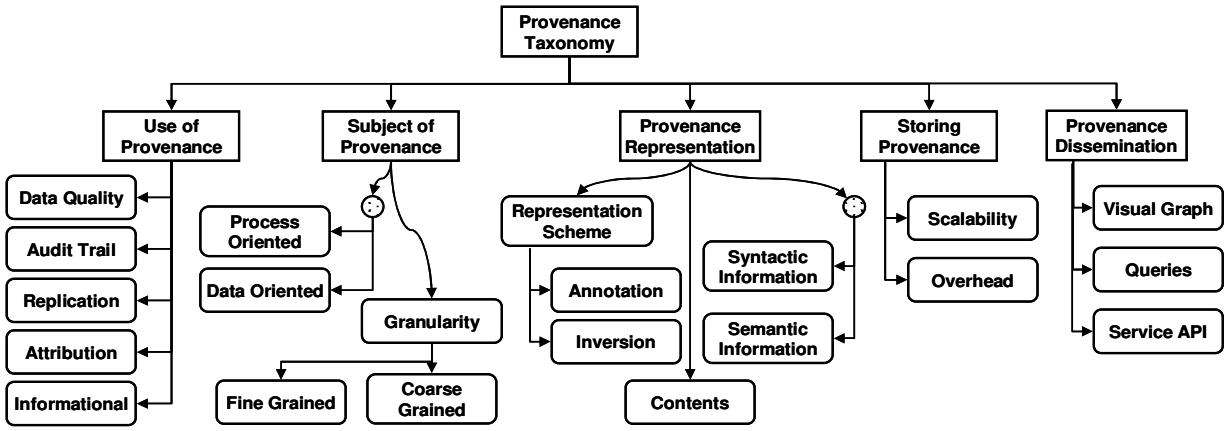


Figure 1 Taxonomy of Provenance

will enable reproduction of the data, or even related publication references [24].

There is no metadata standard for lineage representation across disciplines, and due to their diverse needs, it is a challenge for a suitable one to evolve [25]. Many current provenance systems that use annotations have adopted XML for representing the lineage information [11, 18, 25, 26]. Some also capture semantic information within provenance using domain ontologies in languages like RDF and OWL [18, 25]. Ontologies precisely express the concepts and relationships used in the provenance and provide good contextual information.

2.4 Provenance Storage

Provenance information can grow to be larger than the data it describes if the data is fine-grained and provenance information rich. So the manner in which the provenance metadata is stored is important to its *scalability*. The inversion method discussed in section 2.3 is arguably more scalable than using annotations [19]. However, one can reduce storage needs in the annotation method by recording just the immediately preceding transformation step that creates the data and recursively inspecting the provenance information of those ancestors for the complete derivation history.

Provenance can be tightly coupled to the data it describes and located in the same data storage system or even be embedded within the data file, as done in the headers of NASA Flexible Image Transport System files. Such approaches can ease maintaining the integrity of provenance, but make it harder to publish and search just the provenance. Provenance can also be stored with other metadata or simply by itself [26]. In maintaining provenance, we should consider if it is immutable, or if it can be updated to reflect the current state of its predecessors, or whether it should be versioned [14]. The provenance collection mechanism and its storage repository also determine the trust one places in the provenance and if any mediation service is needed [11].

Management of provenance incurs *costs* for its collection and for its storage. Less frequently used provenance information can be archived to reduce storage overhead or a demand-supply model based on usefulness can retain provenance for those frequently used. If provenance depends on users manually adding annotations instead of automatically collecting it, the burden on the user may prevent complete provenance from being recorded and available in a machine accessible form that has semantic value [18].

2.5 Provenance Dissemination

In order to use provenance, a system should allow rich and diverse means to access it. A common way of disseminating provenance data is through a derivation graph that users can browse and inspect [16, 18, 25, 26]. Users can also search for datasets based on their provenance metadata, such as to locate all datasets generated by a executing a certain workflow. If semantic provenance information is available, these query results can automatically feed input datasets for a workflow at runtime [25]. The derivation history of datasets can be used to replicate data at another site, or update it if a dataset is stale due to changes made to its ancestors [27]. Provenance retrieval APIs can additionally allow users to implement their own mechanism of usage.

3. Survey of Data Provenance Techniques

In our full survey of data provenance [9], we discuss nine major works that, taken together, provide a comprehensive overview of research in this field. In this paper, five works have been selected for discussion. A summary of their characteristics, as defined by the taxonomy, can be found in **Table 1**.

3.1 Chimera

Chimera [27] manages the derivation and analysis of data objects in collaboratory environments and collects provenance in the form of data derivation steps for datasets [17]. Provenance is used for on-demand regeneration of derived data ("virtual data"), comparison of data, and auditing data derivations.

Chimera uses a process oriented model to record provenance. Users construct workflows (called

derivation graphs or DAGs) using a Virtual Data Language (VDL) [17, 27]. The VDL conforms to a schema that represents data products as abstract typed *datasets* and their materialized *replicas*. *Datasets* can be files, tables, and objects of varying granularity, though the prototype supports only files. Computational process templates, called *transformations*, are scripts in the file system and, in future, web services [17]. The parameterized instance of the transformations, called *derivations*, can be connected to form workflows that consume and produce replicas. Upon execution, workflows automatically create *invocation* objects for each derivation in the workflow, annotated with runtime information of the process. Invocation objects are the glue that link input and output data products, and they constitute an annotation scheme for representing the provenance. Semantic information on the dataset derivation is not collected.

The lineage in Chimera is represented in VDL that maps to SQL queries in a relational database, accessed through a virtual data catalog (VDC) service [27]. Metadata can be stored in a single VDC, or distributed over multiple VDC repositories with inter-catalog references to data and processes, to enable scaling. Lineage information can be retrieved from the VDC using queries written in VDL that can, for example, recursively search for derivations that generated a particular dataset. A virtual data browser that uses the VDL queries to interactively access the catalog is proposed [27]. A novel use of provenance in Chimera is to plan and estimate the cost of regenerating datasets. When a dataset has been previously created and it needs to be regenerated (e.g. to create a new replica), its provenance guides the workflow planner in selecting an optimal plan for resource allocation [17, 27].

3.2 myGrid

myGrid provides middleware in support of *in silico* (computational laboratory) experiments in biology, modeled as workflows in a Grid environment [18]. myGrid services include resource discovery, workflow enactment, and metadata and provenance management, which enable integration and present a semantically enhanced information model for bio-informatics.

myGrid is service-oriented and executes workflows written in *XScufl* language using the *Taverna* engine [18]. A provenance log of the workflow enactment contains the services invoked, their parameters, the start and end times, the data products used and derived, and ontology descriptions, and it is automatically recorded when the workflow executes. This process-oriented workflow derivation log is inverted to infer the provenance for the intermediate and final data products. Users need to annotate workflows and services with semantic descriptions to enable this inference and have the semantic metadata carried over to the data products. In addition to contextual and organizational metadata such as owner, project, and experiment hypothesis, ontological terms can also be provided to describe the data and the experiment [8]. XML, HTML, and RDF are used to represent syntactic and semantic provenance metadata using the annotation scheme [14]. The granularity at which provenance can be stored is flexible and is any resource identifiable by an LSID [18].

The myGrid Information Repository (mIR) data service is a central repository built over a relational database to store metadata about experimental components [18]. A number of ways are available for knowledge discovery using provenance. For instance, the semantic provenance information available as RDF can be viewed as a labeled graph using the Haystack semantic web browser [18]. COHSE (Conceptual Open Hypermedia Services Environment), a semantic hyperlink utility, is another tool used to build a semantic web of provenance. Here, semantically annotated provenance logs are interlinked using an ontology reasoning service and displayed as a hyperlinked web page. Provenance information generated during the execution of a workflow can also trigger the rerun of another workflow whose input data parameters it may have updated.

3.3 CMCS

The CMCS project is an informatics toolkit for collaboration and metadata-based data management for multi-scale science [24, 25]. CMCS manages heterogeneous data flows and metadata across multi-disciplinary sciences such as combustion research, supplemented by provenance metadata for establishing the pedigree of data. CMCS uses the Scientific Annotation Middleware (SAM) repository for storing URL referenceable files and collections [25].

CMCS uses an annotation scheme to associate XML metadata properties with the files in SAM and manages them through a Distributed Authoring and Versioning (WebDAV) interface. Files form the level of granularity and all resources such as data objects, processes, web services, and bibliographic records are modeled as files. Dublin Core (DC) verbs like *Has Reference*, *Issued*, and *Is Version Of* are used as XML properties for data files and semantically relate them to their deriving processes through XLink references in SAM [24]. DC elements like *Title* and *Creator*, and user-defined metadata can provide additional context information. Heterogeneous metadata schemas are supported by mapping them to standard DC metadata terms using XSLT translators. Direct association of provenance metadata with the data object makes this a data-oriented model.

There is no facility for automated collection of lineage from a workflow's execution. Data files and their metadata are populated by DAV-aware applications in workflows or manually entered by scientists through a portal interface [25]. Provenance metadata properties

Table 1 Summary of characteristics of surveyed data provenance techniques

	Chimera	myGRID	CMCS	ESSW	Trio
Applied Domain	Physics, Astronomy	Biology	Chemical Sciences	Earth Sciences	None
Workflow Type	Script Based	Service Oriented	Service Oriented	Script Based	Database Query
Use of Provenance	Informational; Audit; Data Replication	Context Information; Re-enactment	Informational; update data	Informational	Informational; up date propagation
Subject	Process	Process	Data	Both	Data
Granularity	Abstract datasets (Presently files)	Abstract resources having LSID	Files	Files	Tuples in Database
Representation Scheme	Virtual Data Language Annotations	XML/RDF Annotations	DublinCore XML Annotations	XML/RDF Annotations	Query Inversion
Semantic Info.	No	Yes	Yes	Proposed	No
Storage Repository/ Backend	Virtual Data Catalog/ Relational DB	mIR repository/ Relational DB	SAM over DAV/ Relational DB	Lineage Server/ Relational DB	Relational DB
User Overhead	User defines derivations; Automated WF trace	User defines Service semantics; Automated WF Trace	Manual: Apps use DAV APIs; Users use portal	Use Libraries to generate provenance	Inverse queries automatically generated
Scalability Addressed	Yes	No	No	Proposed	No
Dissemination	Queries	Semantic browser; Lineage graph	Browser;Queries; GXL/RDF	Browser	SQL/TriQL Queries

can be queried from SAM using generic WebDAV clients. Special portlets allow users to traverse the provenance metadata for a resource as a web page with hyperlinks to related data, or as a labeled graph represented in the Graphics eXchange Language (GXL). The provenance information can also be exported to RDF that semantic agents can use to infer relationships between resources. Provenance metadata that indicate data modification can generate notifications that trigger workflow execution to update dependent data products.

3.4 ESSW

The Earth System Science Workbench (ESSW) [28] is a metadata management and data storage system for earth science researchers. Lineage is a key facet of the metadata created in the workbench, and is used for detecting errors in derived data products and in determining the quality of datasets.

ESSW uses a scripting model for data processing i.e. all data manipulation is done through scripts that wrap existing scientific applications [26]. The sequence of invocation of these scripts by a master workflow script forms a DAG. Data products at the granularity of files are consumed and produced by the scripts, with each data product and script having a uniquely labeled metadata object. As the workflow script invokes individual scripts, these scripts, as part of their execution, compose XML metadata for themselves and the data products they generate. The workflow script links the data flow between successive scripts using their metadata ids to form the lineage trace for all data products, represented as annotations. By chaining the scripts and the data using parent-child links, ESSW is balanced between data and process oriented lineage.

ESSW puts the onus on the script writer to record the metadata and lineage using templates and libraries that are provided. The libraries store metadata objects as files in a web accessible location and the lineage separately in a relational database [26]. Scalability is not currently addressed though it is proposed to federate lineage across organizations. The metadata and lineage information can be navigated as a workflow DAG through a web browser that uses PHP scripts to access the lineage database [28]. Future work includes encoding lineage information semantically as RDF triples to help answer richer queries [26].

3.5 Trio

Cui and Widom [22, 29] trace lineage information for view data in data warehouses. The Trio project [23] leverages some of this work in a proposed database system which has data accuracy and data lineage as inherent components. While data warehouse mining and updation motivates lineage tracking in this project, any e-science system that uses database queries and functions to model workflows and data transformations can apply such techniques.

A database view can be modeled as a query tree that is evaluated bottom-up, starting with leaf operators having tables as inputs and successive parent operators taking as input the result of a child operator [22]. For ASPJ (Aggregate-Select-Project-Join operator) views, it is possible to create an inverse query of the view query that operates on the materialized view, and recursively moves down the query tree to identity the source tables in the leaves that form the view data's lineage [22].

Trio [23] uses this inversion model to automatically determine the source data for tuples created by view

queries. The inverse queries are recorded at the granularity of a view tuple and stored in a special *Lineage* table. This direct association of lineage with tuples makes this a data-oriented provenance scheme. Mechanisms to handle (non-view) tuples created by insert and update queries, and through user-defined functions are yet to be determined. Lineage in Trio is simply the source tuples and the view query that created the view tuple, with no semantic metadata recorded. Scalability is not specifically addressed either. Other than querying the *Lineage* table, some special purpose constructs will be provided for retrieving lineage information through a Trio Query Language (TriQL).

4. Conclusion

In this paper, we presented a taxonomy to understand and compare provenance techniques used in e-science projects. The exercise shows that provenance is still an exploratory field and several open research questions are exposed. Ways to federate provenance information and assert its truthfulness need study for it to be usable across organizations [12]. Evolution of metadata and service interface standards to manage provenance in diverse domains will also contribute to a wider adoption of provenance and promote its sharing [11]. The ability to seamlessly represent provenance of data derived from both workflows and databases can help in its portability. Ways to store provenance about missing or deleted data (phantom lineage [23]) require further consideration. Finally, a deeper understanding of provenance is needed to identify novel ways to leverage it to its full potential.

5. References

[1] J. Brase, "Using Digital Library Techniques - Registration of Scientific Primary Data," in *ECDL*, 2004.

[2] D. G. Clarke and D. M. Clark, "Lineage," in *Elements of Spatial Data Quality*, 1995.

[3] J. L. Romeu, "Data Quality and Pedigree," in *Material Ease*, 1999.

[4] H. V. Jagadish and F. Olken, "Database Management for Life Sciences Research," in *SIGMOD Record*, vol. 33, 2004.

[5] "Access to genetic resources and Benefit-Sharing (ABS) Program," United Nations University, 2003.

[6] P. Buneman, S. Khanna, and W. C. Tan, "Why and Where: A Characterization of Data Provenance," in *ICDT*, 2001.

[7] D. P. Lanter, "Design of a Lineage-Based Meta-Data Base for GIS," in *Cartography and Geographic Information Systems*, vol. 18, 1991.

[8] M. Greenwood, C. Goble, R. Stevens, J. Zhao, M. Addis, D. Marvin, L. Moreau, and T. Oinn, "Provenance of e-Science Experiments - experience from Bioinformatics," in *Proceedings of the UK OST e-Science 2nd AHM*, 2003.

[9] Y. L. Simmhan, B. Plale, and D. Gannon, "A Survey of Data Provenance Techniques," in *Technical Report TR-618*: Computer Science Department, Indiana University, 2005.

[10] R. Bose and J. Frew, "Lineage retrieval for scientific data processing: a survey," in *ACM Comput. Surv.*, vol. 37, 2005.

[11] S. Miles, P. Groth, M. Branco, and L. Moreau, "The requirements of recording and using provenance in e-Science experiments," in *Technical Report, Electronics and Computer Science, University of Southampton*, 2005.

[12] D. Pearson, "Presentation on Grid Data Requirements Scoping Metadata & Provenance," in *Workshop on Data Derivation and Provenance, Chicago*, 2002.

[13] G. Cameron, "Provenance and Pragmatics," in *Workshop on Data Provenance and Annotation, Edinburgh*, 2003.

[14] C. Goble, "Position Statement: Musings on Provenance, Workflow and (Semantic Web) Annotations for Bioinformatics," in *Workshop on Data Derivation and Provenance, Chicago*, 2002.

[15] P. P. da Silva, D. L. McGuinness, and R. McCool, "Knowledge Provenance Infrastructure," in *IEEE Data Engineering Bulletin*, vol. 26, 2003.

[16] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita, "Improving Data Cleaning Quality Using a Data Lineage Facility," in *DMDW*, 2001.

[17] I. T. Foster, J. S. Vöckler, M. Wilde, and Y. Zhao, "The Virtual Data Grid: A New Model and Architecture for Data-Intensive Collaboration," in *CIDR*, 2003.

[18] J. Zhao, C. A. Goble, R. Stevens, and S. Bechhofer, "Semantically Linking and Browsing Provenance Logs for E-science," in *ICSNW*, 2004.

[19] A. Woodruff and M. Stonebraker, "Supporting Fine-grained Data Lineage in a Database Visualization Environment," in *ICDE*, 1997.

[20] B. Plale, D. Gannon, D. Reed, S. Graves, K. Droegemeier, B. Wilhelmson, and M. Ramamurthy, "Towards Dynamically Adaptive Weather Analysis and Forecasting in LEAD," in *ICCS workshop on Dynamic Data Driven Applications*, 2005.

[21] D. Bhagwat, L. Chiticariu, W. C. Tan, and G. Vijayvargiya, "An Annotation Management System for Relational Databases," in *VLDB*, 2004.

[22] Y. Cui and J. Widom, "Practical Lineage Tracing in Data Warehouses," in *ICDE*, 2000.

[23] J. Widom, "Trio: A System for Integrated Management of Data, Accuracy, and Lineage," in *CIDR*, 2005.

[24] C. Pancerella, J. Hewson, W. Koegler, D. Leahy, M. Lee, L. Rahn, C. Yang, J. D. Myers, B. Didier, R. McCoy, K. Schuchardt, E. Stephan, T. Windus, K. Amin, S. Bittner, C. Lansing, M. Minkoff, S. Nijsure, G. v. Laszewski, R. Pinzon, B. Ruscic, Al Wagner, B. Wang, W. Pitz, Y. L. Ho, D. Montoya, L. Xu, T. C. Allison, W. H. Green, Jr, and M. Frenklach, "Metadata in the collaboratory for multi-scale chemical science," in *Dublin Core Conference*, 2003.

[25] J. Myers, C. Pancerella, C. Lansing, K. Schuchardt, and B. Didier, "Multi-Scale Science, Supporting Emerging Practice with Semantically Derived Provenance," in *ISWC workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, 2003.

[26] R. Bose and J. Frew, "Composing Lineage Metadata with XML for Custom Satellite-Derived Data Products," in *SSDBM*, 2004.

[27] I. T. Foster, J.-S. Vöckler, M. Wilde, and Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation," in *SSDBM*, 2002.

[28] J. Frew and R. Bose, "Earth System Science Workbench: A Data Management Infrastructure for Earth Science Products," in *SSDBM*, 2001.

[29] Y. Cui and J. Widom, "Lineage tracing for general data warehouse transformations," in *VLDB Journal*, vol. 12, 2003.

A Notation and System for Expressing and Executing Cleanly Typed Workflows on Messy Scientific Data

Yong Zhao[1] **Jed Dobson**[2] **Ian Foster**[1,3] **Luc Moreau**[4] **Michael Wilde**[3]

[1] Department of Computer Science, University of Chicago, Chicago, IL 60637, U.S.A.
[2] Department of Psychology, Dartmouth College, Hanover, NH 03755, U.S.A.
[3] Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, U.S.A.
[4] School of Electronics and Computer Science, University of Southampton, Southampton, U.K.

Abstract

The description, composition, and execution of even logically simple scientific workflows are often complicated by the need to deal with "messy" issues like heterogeneous storage formats and ad-hoc file system structures. We show how these difficulties can be overcome via a typed, compositional workflow notation within which issues of physical representation are cleanly separated from logical typing, and by the implementation of this notation within the context of a powerful runtime system that supports distributed execution. The resulting notation and system are capable both of expressing complex workflows in a simple, compact form, and of enacting those workflows in distributed environments. We apply our technique to cognitive neuroscience workflows that analyze functional MRI image data, and demonstrate significant reductions in code size relative to other approaches.

1 Introduction

When constructing workflows that operate on large and complex datasets, the ability to describe and introspect on the types of both datasets and workflow components can be invaluable, enabling discovery, type checking, composition, and iteration over compound datasets.

Such typing should in principle be straightforward, because of the hierarchical structure of most scientific datasets. For example, in the functional Magnetic Resonance Imaging (fMRI) applications used for illustrative purposes in this paper, we find a hierarchical structure of studies, groups, subjects, experimental runs, and images (see Figure 1). A typical application might build a new study by applying a program to each image in each run for each subject in each group in a study.

Unfortunately, we find that such clean *logical* structures are typically represented in terms of messy *physical* constructs (e.g., metadata encoded in directory and file names) employed in ad-hoc ways. For example, the fMRI physical representation is a nested directory structure, with ultimately a single 3D image ("volume") represented by two files located in the same directory, distinguished only by file name suffix (Figure 1).

Such messy physical representations make program development, composition, and execution unnecessarily difficult. While we can incorporate knowledge of file system layouts and file formats into application programs and scripts, the resulting code is hard to write and read, cannot easily be adapted to different representations, and is not clearly typed.

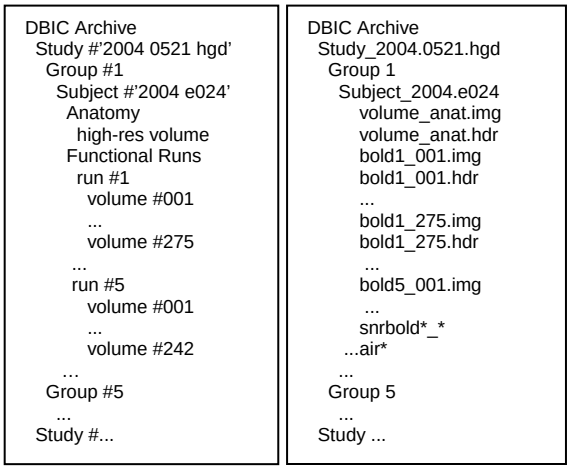


Figure 1: fMRI structure, logical (left) & physical (right)

We have previously proposed that these concerns be addressed by separating abstract structure and physical representation [1]. (Woolf et al. [2] have recently proposed similar ideas.) We describe here the design, implementation, and evaluation of a notation that achieve this separation.

We call this notation a *virtual data language* (VDL) because its declarative structure allows datasets to be defined prior to their generation and without regard to their location and representation. For example, given a VDL procedure "Run Y=foo_run(Run X)" that builds a new run *Y* by applying a program 'foo' to each image in run *X* (*X* and *Y* being dataset variables of type *Run*), we can specify via the statement "run2=foo_run(run1)" that dataset "run2" was (or, alternatively, will be) derived from dataset "run1." Independence from location and

representation is achieved via the use of XML Dataset Typing and Mapping (XDTM) [3] mechanisms, which allow the types of datasets and procedures to be defined abstractly, in terms of XML Schema. Separate *mapping descriptors* then define how such abstract data structures translate to physical representations. Such descriptors specify, for example, how to access the physical files associated with "run1" and "run2."

VDL's declarative and typed structure makes it easy to define increasingly complex procedures via composition. For example, a procedure "Subject Y = foo_subject(Subject X)" might apply "foo_run" to each run in a supplied subject. The repeated application of such compositional forms can ultimately define large directed acyclic graphs (DAGs) comprising thousands or even millions of calls to "atomic transformations" that each operate on just one or two image files.

The expansion of dataset definitions expressed in VDL into DAGs, and the execution of these DAGs as workflows in uni- or multi-processor environments, is the task of an underlying *virtual data system* (VDS).

We have applied our techniques to fMRI data analysis problems [4]. We have modeled a variety of dataset types (and their corresponding physical representations) and constructed and executed numerous computational procedures and workflows that operate on those datasets. Quantitative studies of code size suggest that our VDL and VDS facilitate workflow expression, and hence improve productivity.

We summarize the contributions of this paper as follows:

(1) the design of a practical workflow notation and system that separate logical and physical representation to allow for the construction of complex workflows on messy data using cleanly typed computational procedures;
(2) solutions to practical problems that arise when implementing such a notation within the context of a distributed system within which datasets may be persistent or transient, and both replicated and distributed; and
(3) a demonstration and evaluation of the technology via the encoding and execution of large fMRI workflows in a distributed environment.

The rest of the paper is as follows. In Section 2, we review related work. In Section 3, we introduce the XDTM model and in Section 4 we describe VDL, using an fMRI application for illustration. In Section 5 we describe our implementation, and in Section 6 we conclude with an assessment of results and approach.

2 Related Work

The Data Format Description Language (DFDL) [5], like XDTM, uses XML Schema to describe abstract data models that specify data structures independent from their physical representations. DFDL is concerned with describing legacy data files and complex binary formats, while XDTM focuses on describing data that spans files and directories. Thus, the two systems can potentially be used together.

XPDL [6], BPEL, and WSDL also use XML Schema to describe data or message types, but assume that data is represented in XML; in contrast, XDTM can describe "messy" real-world data. Ptolemy [7] and Kepler [8] provide a static typing system; Taverna [9] and Triana [10] do not mandate typing. The ability to map logical types from/to physical representations is not provided by these languages and systems.

When composing programs into workflows, we must often convert logical types and/or physical representations to make data accessible to downstream programs. XPDL employs scripting languages such as JavaScript to select subcomponents of a data type, and BPEL uses XPath expressions in *Assign* statements for data conversion. Our VDL permits the declarative specification of a rich set of data conversion operations on composite data structures and substructures.

BPEL, YAWL, Taverna, and Triana emphasize web service invocation, while Ptolemy, Kepler, and XPDL are concerned primarily with composing applications. XDTM defines an abstract transformation interface that is agnostic to the procedure invoked, and its binding mechanism provides the flexibility to invoke either web services or applications as needed.

VDL's focus on DAGs limits the range of programs that can be expressed relative to some other systems. However, we emphasize that workflows similar to those presented here are extremely common in scientific computing, in domains including astronomy, bioinformatics, and geographical information systems. VDL can be extended with conditional constructs (for example) if required, but we have not found such extensions necessary to date.

Many workflow languages allow sequential, parallel, and recursive patterns, but do not directly support iteration. Taverna relies on its workflow engine to run a process multiple times when a collection is passed to a singleton-argument process. Kepler adopts a functional operator 'map' to apply a function that operates on singletons to collections. VDL's typing supports flexible iteration over datasets—and also type checking, composition, and selection.

3 XDTM Overview

In XDTM, a dataset's *logical structure* is specified via a subset of XML Schema, which defines primitive scalar data types such as Boolean, Integer, String, Float, and Date, and also allows for the definition of complex types via the composition of simple and complex types.

A dataset's *physical representation* is defined by a *mapping descriptor*, which describes how each element in the dataset's logical schema is stored in, and fetched

from, physical structures such as directories, files, and database tables. In order to permit reuse for different datasets, mapping descriptors can refer to external parameters for such things as dataset location(s).

In order to access a dataset, we need to know three things: its type schema, its mapping descriptor, and the value(s) of any external parameter(s). These three components can be grouped to form a *dataset handle*.

Note that multiple mappings may be defined for the same logical schema (i.e., for a single logical type). For example, an array of numbers might in different contexts be physically represented as a set of relations, a text file, a spreadsheet, or an XML document.

XDTM defines basic constructs for defining and associating physical representations with XML structures. However, it does not speak to how we write programs that operate on XDTM-defined data: a major focus of the work described here.

4 XDTM-Based Virtual Data Language

Our XDTM-based Virtual Data Language (VDL)—derived loosely from an earlier VDL [11], which dealt solely with untyped files—allows users to define procedures that accept, return, and operate on datasets with type, representation, and location defined by XDTM. We introduce the principal features of VDL via an example from fMRI data analysis.

4.1 Application Example

fMRI datasets are derived by scanning the brains of subjects as they perform cognitive or manual tasks. The raw data for a typical study might consist of three subject groups with 20 subjects per group, five experimental runs per subject, and 300 volume images per run, yielding 90,000 volumes and over 60 GB of data. A fully processed and analyzed study dataset can contain over 1.2 million files. In a typical year at the Dartmouth Brain Imaging Center, about 60 researchers preprocess and analyze about 20 concurrent studies.

Experimental subjects are scanned once to obtain a high-resolution image of their brain anatomy ("anatomical volume"), then scanned with a low-resolution imaging modality at rapid intervals to observe the effects of blood flow from the "BOLD" (blood oxygenated level dependant) signal while performing some task ("functional runs"). These images are pre-processed and subjected to intensive analysis that begins with image processing and concludes with a statistical analysis of correlations between stimuli and neural activity.

4.2 VDL Type System

VDL uses a C-like syntax to represent XML Schema types. (There is a straightforward mapping from this syntax to XML Schema.) For example, the first twelve lines of Figure 2 include the VDL types that we use to represent the data objects of Figure 1. (We discuss the procedures later.) Some corresponding XML Schema type definitions are in Figure 3. A *Volume* contains a 3D image of a volumetric slice of a brain image, represented by an *Image* (voxels) and a *Header* (scanner metadata). As we do not manipulate the contents of those objects directly within this VDL program, we define their types simply as (opaque) *String*. A time series of volumes taken from a functional scan of one subject, doing one task, forms a *Run*. In typical experiments, each *Subject* has multiple input and normalized runs, and anatomical data, *Anat*.

```
type Volume { Image img; Header hdr; }
type Image String;
type Header String;
type Run { Volume v[ ]; }
type Anat Volume;
type Subject { Anat anat; Run run [ ]; Run snrun [ ]; }
type Group { Subject s[ ]; }
type Study { Group g[ ]; }
type Air String;
type AirVector { Air a[ ]; }
type Warp String;
type NormAnat {Anat aVol; Warp aWarp; Volume nHires;}

airsn_subject(
        Subject s, Volume atlas, Air ashrink, Air fshrink ) {
    NormAnat a = anatomical(s.anat, atlas, ashrink);
    Run r, snr;
    foreach r in s.run {
        snr = functional ( r, a, fshrink );
        s.snrun[ name(r) ] = snr;
    }
}

(Run snr) functional( Run r, NormAnat a, Air shrink ) {
    Run yroRun  = reorientRun( r , "y" );
    Run roRun   = reorientRun( yroRun , "x" );
    Volume std  = roRun[0];
    Run rndr    = random_select(roRun, .1); //10% sample
    AirVector rndAirVec =
        align_linearRun(rndr, std, 12, 1000, 1000, [81,3,3]);
    Run reslicedRndr = resliceRun( rndr,rndAirVec,"o","k");
    Volume meanRand = softmean(reslicedRndr, "y", null );
    Air mnQAAir =
        alignlinear(a.nHires, meanRand,6,1000,4, [81,3,3]);
    Volume mnQA = reslice(meanRand, mnQAAir, "o","k");
    Warp boldNormWarp =
        combinewarp(shrink, a.aWarp, mnQAAir);
    Run nr = reslice_warp_run( boldNormWarp, roRun );
    Volume meanAll = strictmean ( nr, "y", null )
    Volume boldMask = binarize( meanAll, "y" );
    snr = gsmoothRun( nr, boldMask, 6, 6, 6);
}
```

Figure 2: VDL Dataset Type and Procedure Examples

Specific output formats involved in processing raw input volumes and runs may include outputs from various image processing tools, such as the automated image registration (AIR) suite. The type *Air* corresponds to one dataset type created by these tools.

4.3 Procedures

Datasets are operated on by *procedures*, which take XDTM data as input, perform computations on those data, and produce XDTM data as output. An *atomic* procedure defines an interface to an executable program or service (more on this below); a *compound procedure* composes calls to atomic procedures, compound procedures, and/or foreach statements.

A VDL procedure can be viewed as a *named workflow template*. It defines a *workflow* comprising either a single node (atomic procedure) or multiple nodes (compound procedure). It is a *template* in that its arguments are formal not actual parameters; a call to a VDL procedure instantiates those arguments to define a concrete workflow. Shared variables in the body of a compound procedure specify data dependencies and thus the directed arcs for the DAG corresponding to the compound procedure's workflow.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
targetNamespace="http://www.fmri.org/schema/airsn.xsd"
        xmlns="http://www.fmri.org/schema/airsn.xsd"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">

 <xs:simpleType name="Image">
  <xs:restriction base="xs:string"/>
 </xs:simpleType>

 <xs:simpleType name="Header">
  <xs:restriction base="xs:string"/>
 </xs:simpleType>

 <xs:complexType name="Volume">
  <xs:sequence>
   <xs:element name="img" type="Image"/>
   <xs:element name="hdr" type="Header"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="Run">
  <xs:sequence minOccurs="0 maxOccurs="unbounded">
   <xs:element name="v" type="Volume"/>
  </xs:sequence>
 </xs:complexType>

</xs:schema>
```

Figure 3: Type Definitions in XML Schema

We use as our illustrative example a workflow, AIRSN, that performs *spatial normalization* for pre-processing raw fMRI data prior to analysis. AIRSN normalizes sets of time series of 3D volumes to a standardized coordinate system and applies motion correction and Gaussian smoothing. Figures 4 and 5 show two views of the most data-intensive segment of the AIRSN workflow, which processes the data from the functional runs of a study. Figure 4 is a high-level representation in which each oval represents an operation performed on an entire Run. Figure 5 expands the workflow to the Volume level, for a dataset of 10 functional volumes. (The alert reader may note that the random_select call is missing; this is an unimportant artefact.) In realistic fMRI science runs, Runs might include hundreds or thousands of volumes.

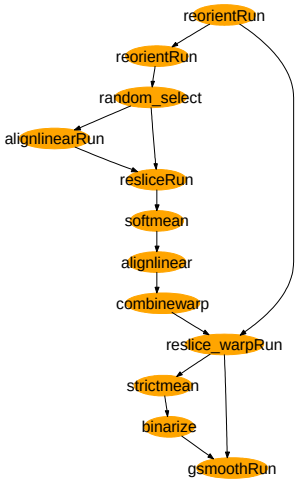


Figure 4: AIRSN workflow high-level representation

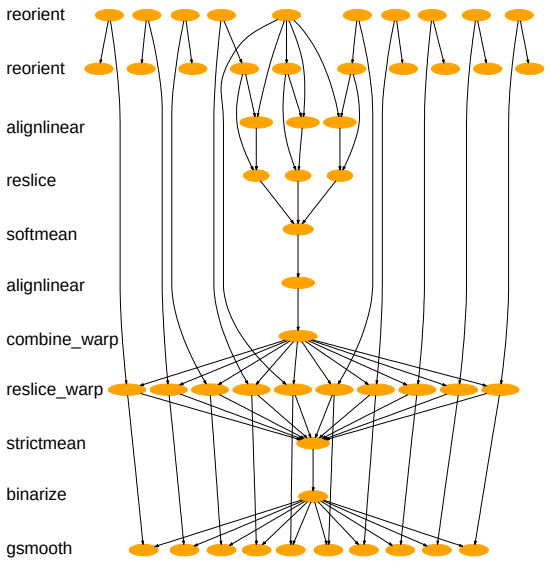


Figure 5: AIRSN workflow expanded to show all atomic file operations, for a 10 volume run

We present a subset of the VDL for AIRSN in Figure 2. The procedure *functional* expresses the steps in Figure 4; *airsn_subject* calls both *functional* and procedure *anatomical* (not shown) to process a *Subject*.

The VDL *foreach* statement allows programs to apply an operation to all components of a compound data object. For example, *airsn_subject* creates in the *Subject* dataset a new spatially normalized *Run* for each raw *Run*. Such procedures define how the workflow is expanded as in Figure 5.

To apply a VDL procedure to a specific physical dataset, we simply pass a reference to that dataset as an

actual parameter. The resulting call will execute correctly regardless of the physical representation of a passed dataset (assuming that the dataset and procedure have matching logical types). Internally, dataset references take the form of handles, which, as described in Section 3, contain type, mapping, and location information. As in languages in which every variable is an object reference, handles are never seen by the user.

4.4 Invoking Programs and Services

A workflow such as Figure 2 must ultimately invoke external executable programs and/or Web Services. VDL *atomic procedures* define the necessary interfaces, specifying the name of the program or service to be invoked, how to set up its execution environment, how program arguments or service messages should be mapped from and to VDL procedure arguments, and what physical data objects need to be moved to and from remote execution sites.

```
(Air out) alignlinear(Volume std, Volume v,
                      Int m, Int t1, Int t2, Int s[ ] ) {
   argument = out;
   argument = get_member(std, hdr);
   argument = get_member (v, hdr);
   argument = "-m " m;
   argument = "-t1" t1;
   argument = "-t2" t2;
   argument = "-s " s[0] s[1] s[2];
}
```

Figure 6 Program Invocation

For example, the procedure *alignlinear* called in Figure 2 defines a VDL interface to the AIR utility of the same name. There are two important things to understand about this definition. First, the VDS ensures that if this call is executed on a remote computer (as is usually the case in a distributed environment), the physical representations of datasets passed as input arguments are transferred to the remote site, thus ensuring that the executable can access the required physical files. In the case of output data (e.g., "Air a"), the physical data is left on the remote site, registered in a replica location service, and optionally copied to another specified site to create an additional replica (which often serves as an archival copy).

Second, the statements in the body assemble the command to invoke the program, so that for example the VDL call:

```
Air a = alignlinear(t1a, t3, 12, 1000, 1000, [81 3 3])
```

requests the execution of the following command:

```
alignlinear a.air t1a.hdr t3.hdr -m 12 \
                -t1 1000 -t2 1000 -s 81 3 3
```

Alternative atomic procedures can be provided to specify Web Service interfaces to the utilities. These alternative procedures would implement the same procedure prototype, but provide a different body.

5 Implementation

We have developed a prototype system that can process VDL type definitions and mappings, convert a typed workflow definition into an executable DAG, expand DAG nodes dynamically to process sub-components of a compound dataset, and submit and execute the resulting DAG in a Grid environment. The separation of dataset type and physical representation that we achieve with VDL can facilitate various runtime optimizations and graph rewriting operations [12].

Our prototype does not yet include a parser for the syntax presented here. However, the prototype does implement the runtime operations needed to support typed VDL dataset processing and execution, which is the principal technical challenge of implementing VDL. We have also verified that we can invoke equivalent services and applications from the same VDL.

The prototype extends an earlier VDS implementation with features to handle data typing and mapping. We use the VDS graph traversal mechanism to generate an abstract DAG in which transformations are not yet tied to specific applications or services, and data objects are not yet bound to specific locations and physical representations. The extended VDS also enhances the DAG representation by introducing "foreach" nodes (in addition to the existing "atomic" nodes) to represent *foreach* statements in a VDL procedure. These nodes are expanded at runtime (see Section 5.2), thus enabling datasets to have a dynamically determined size.

The abstract DAG is concretized by a Grid planner called Euryale, which produces a concrete DAG that, for each node in the input abstract DAG, performs the following steps. (See Sections 5.1 and 5.2 for details on how Euryale performs data mapping during these steps, and expands *foreach* statements, respectively.)

1. *Preprocess*:
   if (atomic procedure node) {
     identify node inputs and outputs;
     choose Grid site that meets job requirements;
     locate and transfer inputs to that site;
   }
   else if (foreach node)
     expand foreach statement(s) into sub-dag(s);
2. *Execute*: Submit job or sub-DAG; wait for it to execute.
3. *Postprocess*: Check job exit status; transfer and register outputs; cleanup.

The resulting concrete DAG is executed by the DAGman ("DAG manager") tool. DAGman provides many necessary facilities for workflow execution, such as logging, job status monitoring, workflow persistence, and recursive fault recovery. DAGman submits jobs to Grid sites via the Globus GRAM protocol.

5.1 Data Mapping

The Eurayle planner needs to operate on the physical data that lies behind the logical types defined in VDL procedures. Such operations are accessed via a mapping descriptor associated with the dataset, which controls the execution of a *mapping driver* used to map between physical and abstract representations. In general, a mapping driver must implement the functions *create-dataset*, *store-member*, *get-member*, and *get-member-list*. Our prototype employs a table-driven approach to implement a mapping driver for file-system-stored datasets. Each table entry specifies:

name: *the data object name*
pattern: *the pattern used to match file names*
mode: **FILE** *(find matches in directory)*
RLS *(find matches via replica location service),*
ENUM *(dataset content is enumerated)*
content: *used in* **ENUM** *mode to list content*

When mapping an input dataset, this table is consulted, the pattern is used to match a directory or replica location service according to the mode, and the members of the dataset are enumerated in an in-memory structure. This structure is then used to expand *foreach* statements and to set command-line arguments.

For example, recall from Figure 1 that a *Volume* is physically represented as an image/header file pair, and a *Run* as a set of such pairs. Furthermore, multiple Runs may be stored in the same directory, with different Runs distinguished by a prefix and different Volumes by a suffix. To map this representation to the logical *Run* structure, the pattern 'bold*N**' is used to identify all pairs in Run *N* at a specified location. Thus, the mapper, when applied to the following eight files, identifies two runs, one with three *Volumes* (*Run 1*) and the other with one (*Run 2*).

```
bold1_001.img    bold1_001.hdr
bold1_002.img    bold1_002.hdr
bold1_003.img    bold1_003.hdr
bold2_007.img    bold2_007.hdr
```

5.2 Dynamic Node Expansion

A node containing a *foreach* statement must be expanded prior to execution into a set of nodes: one per component of the compound data object specified in the *foreach*. This expansion is performed at runtime: when a *foreach* node is scheduled for execution, the appropriate mapper function is called on the specified dataset to determine its members, and for each member of the dataset identified (e.g., for each *Volume* in a *Run*) a new job is created in a "sub-DAG."

The new sub-DAG is submitted for execution, and the main job waits for the sub-DAG to finish before proceeding. A post-script for the main job takes care of the transfer and registration of all output files, and the collection of those files into the output dataset. This workflow expansion process may itself recurse further if the subcomponents themselves also include *foreach* statements. DAGman provides workflow persistence even in the face of system failures during recursion.

5.3 Optimizations and Graph Transformation

Since dataset mapping and node expansion are carried out at run time, we can use graph transformations to apply optimization strategies. For example, in the AIRSN workflow, some processes, such as the *reorient* of a single *Volume*, only take a few seconds to run. It is inefficient to schedule a distinct process for each *Volume* in a *Run*. Rather, we can combine multiple such processes to run as a single job, thus reducing scheduling and queuing overhead.

As a second example, the *softmean* procedure computes the mean of all *Volume*s in a *Run*. For a dataset with large number of *Volume*s, this stage is a bottleneck as no parallelism is engaged. There is also a practical issue: the executable takes all *Volume* filenames as command line arguments, which can exceed limits defined by the Condor and UNIX shell tools used within our VDS implementation. Thus, we transform this node into a tree in which leaf nodes compute over subsets of the dataset. The process repeats until we get a single output. The shape of this tree can be tuned according to available computing nodes and dataset sizes to achieve optimal parallelism and avoid command-line length limitations.

6 Evaluation

We have used our prototype system to execute a range of fMRI workflows with various input datasets on the Dartmouth Green Grid, which comprises five 12-node clusters. The dataset mapping mechanism allowed us to switch input datasets (e.g., from a Run of 80 volumes to another Run of 120 volumes) without changing either the workflow definition or the execution system. All workflows run correctly and achieve speedup.

The primary focus of our work is to increase productivity [13]. As an approximate measure of this, we compare in Table 1 the lines of code needed to express five different fMRI workflows, coded in our new VDL, with two other approaches, one based on ad-hoc shell scripts ("Script," able to execute only on a single computer) and a second ("Generator") that uses Perl scripts to generate older, "pre-XDTM" VDL.

Table 1: Lines of code with different workflow encodings

Workflow	Script	Generator	VDL
GENATLAS1	49	72	6
GENATLAS2	97	135	10
FILM1	63	134	17
FEAT	84	191	13
AIRSN	215	~400	37

The new programs are smaller and more readable—and also provide for type checking, provenance tracking, parallelism, and distributed execution.

7 Conclusions

We have designed a typed workflow notation and system that allows workflows to be *expressed* in terms of declarative procedures that operate on XML data types and then *executed* on diverse physical representations and on distributed computers. We show that this notation and system can be used to express large amounts of distributed computation easily.

The productivity leverage of this approach is apparent: a small group of developers can define VDL interfaces to the utility packages used in a research domain and then create a library of dataset-iteration functions. This library encapsulates low-level details concerning how data is grouped, transported, catalogued, passed to applications, and collected as results. Other scientists can then use this library to construct workflows without needing to understand details of physical representation, and furthermore are protected by the XDTM type system from forming workflows that are not type compliant. In addition, the data management conventions of a research group can be encoded in XDTM mapping functions, thus making it easier to maintain order in dataset collections that may include tens of millions of files.

We next plan to automate the parsing steps that were performed manually in our prototype, and to create a complete workflow development and execution environment for our XDTM-based VDL. We will also investigate support for services, automation of type coercions between differing physical representations, and recording of provenance for large data collections.

Acknowledgements.
This work was supported by the National Science Foundation GriPhyN Project, grant ITR-800864, the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, and by the National Institutes of Health, grants NS37470 and NS44393. We are grateful to Scott Grafton of the Dartmouth Brain Imaging Center, and to Jens Voeckler, Doug Scheftner, Ewa Deelman, Carl Kesselman, and the entire Virtual Data System team for discussion, guidance, and assistance.

References

[1] Foster, I., Voeckler, J., Wilde, M., Zhao, Y. The Virtual Data Grid: A New Model and Architecture for Data-intensive Collaboration. *Conference on Innovative Data Systems Research*, Asilomar, CA, January 2003.

[2] Woolf, A., Cramer, R., Gutierrez, M., van Dam, K., Kondapalli, S., Latham, S., Lawrence, B., Lowry, R., O'Neill, K., Semantic Integration of File-based Data for Grid Services. *Workshop on Semantic Infrastructure for Grid Computing Applications*, 2005.

[3] Moreau, L., Zhao, Y., Foster, I., Voeckler, J. Wilde, M., XDTM: XML Dataset Typing and Mapping for Specifying Datasets. *European Grid Conference*, 2005.

[4] Van Horn, J.D., Dobson, J., Woodward, J., Wilde, M., Zhao, Y., Voeckler, J., Foster, I. Grid-Based Computing and the Future of Neuroscience Computation, *Methods in Mind*, Cambridge: MIT Press (In Press).

[5] Beckerle, M., Westhead, M. GGF DFDL Primer. Technical report, Global Grid Forum, 2004.

[6] XML Process Definition Language (XPDL) (WFMCTC-1025). Technical report, Workflow Management Coalition, Lighthouse Point, Florida, USA, 2002.

[7] Eker, J., Janneck, J., Lee, E., Liu, J., Liu, X., Ludvig, J., Neuendorffer, S., Sachs, S., Xiong, Y. Taming Heterogeneity – the Ptolemy Approach. *Proceedings of the IEEE*, 91(1):127-144, January 2003.

[8] Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludäscher, B. and Mock, S., Kepler: An Extensible System for Design and Execution of Scientific Workflows. *16th Intl. Conference on Scientific and Statistical Database Management*, 2004.

[9] Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M., Wipat, A., Li, P. Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows. *Bioinformatics Journal*, 20(17):3045-3054, 2004.

[10] Churches, D., Gombas, G., Harrison, A., Maassen, J., Robinson, C., Shields, M., Taylor, I. Wang, I. Programming Scientific and Distributed Workflow with Triana Services. *Concurrency and Computation: Practice and Experience*, 2005 (in press).

[11] Foster, I., Voeckler, J., Wilde, M., Zhao, Y. Chimera: A Virtual Data System for Representing, Querying and Automating Data Derivation. *14th Conference on Scientific and Statistical Database Management*, 2002.

[12] Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Blackburn, K., Lazzarini, A., Arbree, A., Cavanaugh, R., Koranda, S. Mapping Abstract Workflows onto Grid Environments. *Journal of Grid Computing*, 1(1). 2003.

[13] Gray, J., Liu, D., Nieto-Santisteban, M., Szalay, A. Scientific Data Management in the Coming Decade. Microsoft Research, MSR-TR-2005-10. 2005.

A Taxonomy of Scientific Workflow Systems for Grid Computing

Jia Yu and Rajkumar Buyya*
Grid Computing and **D**istributed **S**ystems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
htpp://www.gridbus.org

ABSTRACT
With the advent of Grid and application technologies, scientists and engineers are building more and more complex applications to manage and process large data sets, and execute scientific experiments on distributed resources. Such application scenarios require means for composing and executing complex workflows. Therefore, many efforts have been made towards the development of workflow management systems for Grid computing. In this paper, we propose a taxonomy that characterizes and classifies various approaches for building and executing workflows on Grids. The taxonomy not only highlights the design and engineering similarities and differences of state-of-the-art in Grid workflow systems, but also identifies the areas that need further research.

Keywords
Grid computing, Taxonomy, Scientific workflows.

1. INTRODUCTION
Grids [9] have emerged as a global cyber-infrastructure for the next-generation of e-Science applications, by integrating large-scale, distributed and heterogeneous resources. Scientific communities, such as high-energy physics, gravitational-wave physics, geophysics, astronomy, and bioinformatics, are utilizing Grids to share, manage and process large data sets. In order to support complex scientific experiments, distributed resources such as computational devices, data, applications, and scientific instruments need to be orchestrated while managing workflow operations within Grid environments [15].

Scientific workflow is concerned with the automation of scientific processes in which tasks are structured based on their control and data dependencies. The workflow paradigm for scientific applications on Grids offers several advantages, such as (a) ability to build dynamic applications which orchestrate distributed resources, (b) utilizing resources that are located in a particular domain to increase throughput or reduce execution costs, (c) execution spanning multiple administrative domains to obtain specific processing capabilities, and (d) integration of multiple teams involved in management of different parts of the experiment workflow – thus promoting inter-organizational collaborations.

In the recent past, several Grid workflow systems have been proposed and developed for defining, managing and executing scientific workflows. In order to enhance understanding of the field, we propose a taxonomy that primarily (a) captures architectural styles and (b) identifies design and engineering similarities and differences between them. The taxonomy provides an in-depth understanding of building and executing workflows on Grids. It compares different approaches and also helps users to decide on minimum subset of features required for their systems.

The rest of the paper is organized as follows: Section 2 presents taxonomy that classifies approaches based on major functions and architectural styles of Grid workflow systems. In Section 3, we map the proposed taxonomy onto selected Grid workflow systems and conclude in Section 4.

2. TAXONOMY
The taxonomy characterizes and classifies approaches of scientific workflow systems in the context of Grid computing. It consists of four elements of a Grid workflow management system: (a) workflow design, (b) workflow scheduling, (c) fault tolerance and (d) data movement (see Figure 1). In this section, we look at each element and its taxonomy briefly. A detailed taxonomy and in depth discussion on its mapping can be found in [23].

2.1 Workflow Design
Workflow design determines how workflow components can be defined and composed.

2.1.1 Workflow Structure
A workflow is composed by connecting multiple scientific tasks according to their dependencies. Workflow structure indicates the temporal relationship between there tasks. In general, a workflow can be represented as a *Directed Acyclic Graph (DAG)* or a *non-DAG*.

In DAG-based workflow, workflow structure can be categorized as *sequence*, *parallelism*, and *choice*. Sequence is defined as an ordered series of tasks, with one task starting after a previous task has completed. Parallelism represents tasks which are performed concurrently, rather than serially. In choice structured workflows, a task is selected to execute at run-time when its associated conditions are true. In addition to all structures contained in a DAG-based, a non-DAG workflow also includes *iteration* structure, in which sections of workflow tasks in an iteration block are allowed to be repeated.

*Corresponding author, raj@cs.mu.oz.au

2.1.2 Workflow Model/Specification

Workflow Model (also called workflow specification) defines a workflow including its task definition and structure definition. There are two types of workflow models, namely *abstract* and *concrete*.

In the abstract model, a workflow is described in an abstract form, in which the workflow is specified without referring to specific Grid resources for task execution. In contrast, the concrete model binds workflow tasks to specific resources. Given the dynamic nature of the Grid environment, it is more suitable for users to define workflow applications in the abstract model. A full or partial concrete model can be generated just before or during workflow execution, according to the current status of resources.

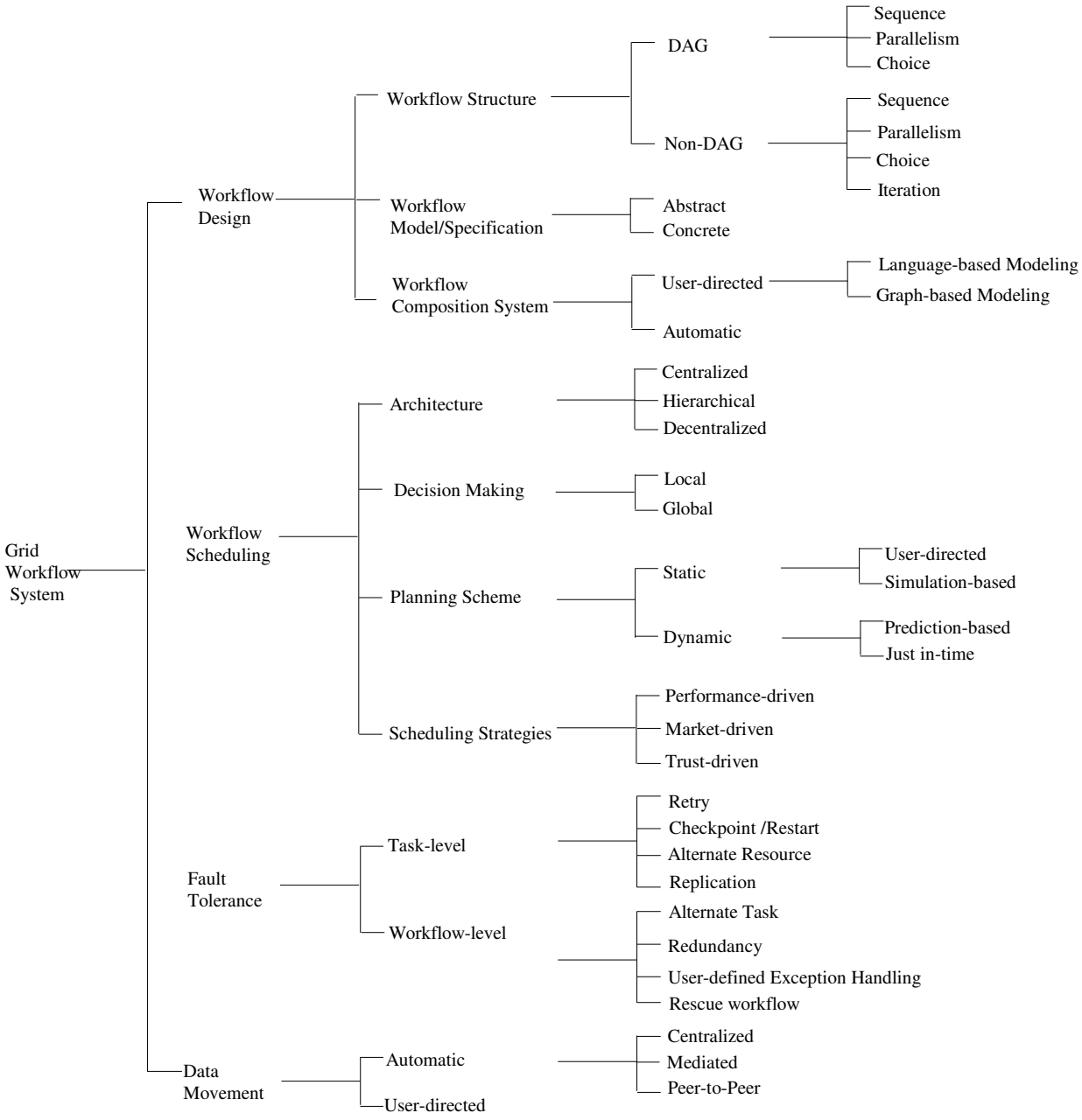


Figure 1. A taxonomy of scientific workflow systems for Grid computing.

2.1.3 Workflow Composition System

Workflow composition systems are designed for enabling users to assemble components into workflows. They need to provide a high level view for the construction of Grid workflow applications and hide the complexity of

underlying Grid systems. The composition systems are categorized as *user-directed* and *automatic.*

User-directed systems require users to edit workflows directly. In general, users can use workflow languages such as Extensible Markup Language (XML) [21] for *language-based modeling* and the tools such as Kepler [2] for *graph-based modeling* to compose workflows. Compared with language-based modeling, Graphical representation is very intuitive and can be handled easily even by a non-expert user. However, workflow languages are more appropriate for storing and transfer whereas the graphical representation is required to be converted into other form for such manipulation.

Automatic systems generate workflows for users automatically according to their higher level requirements, such as data products and initial input values. Compared with user-directed systems, automatic systems are ideal for large scale workflows which are very time consuming to compose manually. However, the automatic composition of application components is challenging because it is difficult to capture the functionality of components and data types used by the components.

2.2 Workflow Scheduling

Workflow scheduling focuses on mapping and managing the execution of workflow tasks on shared resources that are not directly under the control of workflow systems.

2.2.1 Scheduling Architecture

The architecture of the scheduling infrastructure is very important for the scalability, autonomy, quality and performance of the system [11]. Three major categories of workflow scheduling architecture are *centralized*, *hierarchical* and *decentralized* scheduling schemes.

In the centralized workflow enactment environment, one central scheduler makes scheduling decisions for all tasks in the workflow. For hierarchical scheduling, there is a central manager and multiple lower-level sub-workflow schedulers. This central manager is responsible for controlling the workflow execution and assigning sub-workflows to the lower-level schedulers. In contrast with the centralized and hierarchical schemes, there are multiple schedulers without any central controller in decentralized scheduling. Every scheduler can communicate each other and schedule a sub-workflow to another scheduler with lower load.

It is believed that the centralized scheme can produce efficient schedules because it has all necessary information about all tasks in workflows. However, it is not scalable with respect to the number of tasks, the classes and number of Grid resources that are generally autonomous. The major advantage of using the hierarchical architecture is that different scheduling policies can be deployed in the central manager and lower-level schedulers [11]. However, the failure of the central manager will result in entire system failure. Decentralized scheduling is more scalable but faces more challenges to generate optimal solutions for overall workflow performance.

2.2.2 Decision Making

It is difficult to find a single best solution for mapping workflows onto resources for all workflow applications, since applications can have very different characteristics. It depends to some degree on the application models to be scheduled. In general, decisions about mapping tasks in a workflow onto resources can be based on the information of the current task or of the entire workflow. Scheduling decisions made with reference to just the task or sub-workflow at hand are called *local decisions* while scheduling decision made with reference to the whole workflow are called *global decisions* [5].

It is believed that global decision based scheduling can provide better overall results, since local decision scheduling only takes one task or sub-workflow into account. However, it also takes much more time in scheduling decision making. The overhead produced by global decision based scheduling will reduce the overall benefit and can even exceed the benefits it will produce. Therefore, the decision making of workflow scheduling should consider both overall execution time and scheduling time.

2.2.3 Planning Scheme

Schemes for translating abstract models to concrete models can be categorized into either *static* or *dynamic*. In a static scheme, concrete models have to be generated before the execution, according to current information about the execution environment, and the dynamically changing state of the resources is not taken into account. In contrast, a dynamic scheme uses both dynamic and static information about resources in order to make scheduling decisions at run-time.

Static schemes can be classified as *user-directed* or *simulation-based*. In user-directed scheduling, users emulate the scheduling process and make resource mapping decisions according to their knowledge, preference and/or performance criteria. In simulation-based scheduling, a 'best' schedule is achieved by simulating task execution on a given set of resources before a workflow starts execution. The simulation can be processed based on static information or the result of performance estimation.

Dynamic schemes include *prediction-based* and *just in-time* scheduling. Prediction-based dynamic scheduling uses dynamic information in conjunction with some results based on prediction. It is similar to simulation-based static scheduling, in which the scheduler is required to predict the performance of task execution on resources and generate a near optimal schedule for the task before it starts the execution. However, it changes the initial schedule dynamically during the execution. Rather than making a

schedule prior to scheduling, just in-time scheduling only makes a scheduling decision at the time of task execution.

Planning ahead in Grid environments may produce a poor schedule, since it is a dynamic environment where utilization and the availability of resources varies over time and a better resource can join at any time. Moreover, it is not easy to accurately predict execution time of all application components on Grid resources. However, as the technology of advance reservation for various resources improve, it is believed that the role of static and prediction-based planning will increase [5] .

2.2.4 Scheduling Strategy

In general, scheduling workflow applications in a distributed system is an NP-complete problem [8]. Many heuristics have been developed to obtain near-optimal solutions to match users' QoS constraints such as deadline and budget.

Performance-driven strategies try to find a mapping of workflow tasks onto resources that achieves optimal execution performance such as minimum overall execution time. *Market-driven* strategies [10] employ market models to manage resource allocation for processing workflow tasks. Workflow schedulers with a market-driven strategy act as consumers buying services from resource providers and pay some form of electronic currency for executing tasks in workflows. Unlike performance-driven strategies, market-driven schedulers may choose a resource with later deadline if its usage price is cheaper.

Recently *trust-driven* scheduling approaches [18] in distributed systems are emerging. Trust-driven schedulers select resources based on their trust properties such as security policy, accumulated reputation, self-defense capability, attack history, and site vulnerability. By using trust-driven approaches, the overall reliability of workflow systems can be increased by reducing the chance of selecting malicious hosts and disreputable resources.

2.3 Fault Tolerance

In Grid environments, workflow execution failures can occur for various reasons such as network failure, overloaded resource conditions, or non-availability of required software components. Thus, Grid workflow management systems should be able to identify and handle failures and support reliable execution in the presence of concurrency and failures. Workflow failure handling techniques are classified as *task-level* and *workflow-level* [12]. Task-level techniques mask the effects of the execution failure of tasks in the workflow, while workflow-level techniques manipulate the workflow structure such as execution flow to deal with erroneous conditions.

Task-level techniques have been widely studied in parallel and distributed systems. They can be classified as *retry*, *alternate resource*, *checkpoint/restart* and *replication*. The retry technique is the simplest failure recovery technique, as it simply tries to execute the same task on the same resource after failure. The alternate resource technique submits failed task to another resource. The checkpoint/restart technique moves failed tasks transparently to other resources, so that the task can continue its execution from the point of failure. The replication technique runs the same task simultaneously on different Grid resources to ensure task execution provided that at least one of the replicas does not fail.

Workflow-level techniques are classified as *alternate task*, *redundancy*, *user-defined exception handling* and *rescue workflow*. The alternate task technique executes another implementation of a certain task if the previous one failed, while the redundancy technique executes multiple alternative tasks simultaneously. User-defined exception handling allows users to specify a special treatment for a certain failure of a task in the workflow. The rescue workflow technique generates a rescue workflow, which records information about failed tasks, during the first workflow execution. The rescue workflow is used for later submission.

2.4 Intermediate Data Movement

For Grid workflow applications, the input files of tasks need to be staged to a remote site before processing tasks. Similarly, output files may be required by their children tasks which are processed on other resources. Therefore, intermediate data has to be staged out to corresponding Grid sites. Some systems require users to manage intermediate data transfer in the workflow specification (*user-directed* approach), while some systems provide *automatic* mechanisms to transfer intermediate data.

We classify approaches of automatic intermediate data movement as *centralized*, *mediated* and *peer-to-peer*. A centralized approach transfers intermediate data between resources via a central point. In a mediated approach rather than using a central point, the locations of the intermediate data are managed by a distributed data management system. A peer-to-peer approach transfers data between processing resources.

In general, centralized approaches are easily implemented and suit workflow applications in which large-scale data flow is not required. Mediated approaches are more scalable and suitable for applications which need to keep intermediate data for later use. Since data is transmitted from a source resource to a destination resource directly, without involving any third-party service, peer-to-peer approaches save transmission time significantly and reduce the bottleneck caused by the centralized and mediated approaches. Thus, the peer-to-peer approach is more suitable for large-scale intermediate data transfer. However, there are more difficulties in deployment because it requires a Grid site to be capable of providing both data management and movement service.

3. GRID WORKFLOW SYSTEM SURVEY

A mapping of taxonomy to several existing Grid workflow systems is shown in Table 1. The detailed discussion on these systems along with identification of areas that need further work can be found in [23].

Table 1. Taxonomy mapping to Grid workflow systems.

Project	Workflow Design			Workflow Scheduling				Fault-tolerance	Data Movement
	Structure	Model	Composition System	Architecture	Decision Making	Planning Scheme	Strategies		
DAGMan [19]	DAG	Abstract	User-directed -Language-based	Centralized	Local	Dynamic -Just in-time	Performance-driven	Task Level -Migration -Retrying Workflow Level -Rescue workflow	User-directed
Pegasus [6]	DAG	Abstract	User-directed -Language-based Automatic	Centralized	Local Global	Static -user-directed Dynamic -Just in-time	Performance-driven	Based on DAGMan	Mediated
Triana [20]	Non-DAG	Abstract	User-directed -Graph-based	Decentralized	Local	Dynamic -Just in-time	Performance-driven	Based on GAT manger	Peer-to-Peer
ICENI [16]	Non-DAG	Abstract	User-directed -Language-based -Graph-based	Centralized	Global	Dynamic -Prediction-based	Performance-driven Market-driven	Based on ICENI middleware	Mediated
Taverna [17]	DAG	Abstract Concrete	User-directed -Language-based -Graph-based	Centralized	Local	Dynamic -Just in-time	Performance-driven	Task Level -Retry -Alternate Resource	Centralized
GrADS [3]	DAG	Abstract	User-directed -Language-based	Centralized	Local Global	Dynamic -Prediction-based	Performance-driven	Task Level in rescheduling work in GrADS, but not in workflows.	Peer-to-Peer
GridFlow [4]	DAG	Abstract	User-directed -Graph-based -Language-based	Hierarchical	Local	Static -Simulation-based	Performance-driven	Task Level -Alternate resource	Peer-to-Peer
UNICORE [1]	Non-DAG	Concrete	User-directed -Graph-based	Centralized	User-defined*	Static -User-directed	User-defined*	Based on UNICORE middleware	Mediated
Gridbus workflow [22]	DAG	Abstract Concrete	User-directed -Language-based	Hierarchical	Local	Static -User-directed Dynamic -Just in-time	Market-driven	Task Level -Alternate resource	Centralized Peer-to-Peer
Askalon [7]	Non-DAG	Abstract	User-directed -Graph-based -Language-based	Decentralized	Global	Dynamic -Just in-time -Prediction-based	Performance-driven Market-driven	Task Level -Retry -Alternate resource Workflow level -Rescue workflow	Centralized User-directed
Karajan [13]	Non-DAG	Abstract	User-directed -Graph-based -Language-based	Centralized	User-defined*			Task Level -Retry -Alternate resource -checkpoint/restart Workflow Level -User-defined exception handling	User-directed
Kepler [14]	Non-DAG	Abstract Concrete	User-directed -Graph-based	Centralized	User-defined*			Task Level - Alternate resource Workflow Level - User-defined exception handling - Workflow rescue	Centralized Mediated Peer-to-Peer

* **user-defined** – the architecture of the system has been explicitly designed for user extension.

4. CONCLUSION

We have presented a taxonomy for Grid workflow systems. The taxonomy focuses on workflow design, workflow scheduling, fault management and data movement. We also survey some workflow management systems for Grid computing and classify them into different categories using the taxonomy. This paper thus helps to understand key

workflow management approaches and identify possible future enhancements.

5. ACKNOWLEDGMENTS

We would like to acknowledge all developers of the workflow management systems described in the paper. We thank Chee Shin Yeo, Hussein Gibbins, Anthony Sulistio, Srikumar Venugopal, Tianchi Ma, Sushant Goel, and Baden Hughes (Melbourne University, Australia), Rob Gray (Monash University, Australia), Wolfram Schiffmann (FernUniversitaet in Hagen, Germany), Ivona Brandic (University of Vienna, Austria), Soonwook Hwang (National Institute of Informatics, Japan), Ewa Deelman (University of Southern California, USA), Chris Mattmann (NASA Jet Propulsion Laboratory, USA), Henan Zhao (University of Manchester, UK), Bertram Ludaescher (University of California, Davis), Thomas Fahringer (University of Innsbruck, Austria), Gregor von Laszewski (Argonne National Laboratory, USA), Ken Kennedy, Anirban Mandal, and Chuck Koelbel (Rice University, USA) for their comments on this paper. We thank anonymous reviewers for their constructive comments. This work is partially supported through the Australian Research Council (ARC) Discovery Project grant and Storage Technology Corporation sponsorship of Grid Fellowship.

6. REFERENCES

[1] J. Almond and D. Snelling. UNICORE: Secure and Uniform Access to Distributed Resources via the World Wide Web. *White Paper*, October 1998,

[2] I. Altintas et al. A Framework for the Design and Reuse of Grid Workflows, *International Workshop on Scientific Applications on Grid Computing (SAG'04)*, LNCS 3458, Springer, 2005

[3] F. Berman et al. The GrADS Project: Software Support for High-Level Grid Application Development. *International Journal of High Performance Computing Applications(JHPCA)*, 15(4):327-344, SAGE Publications Inc., London, UK, Winter 2001.

[4] J. Cao et al. GridFlow:Workflow Management for Grid Computing. In *3rd International Symposium on Cluster Computing and the Grid (CCGrid)*, Tokyo, Japan, IEEE CS Press, Los Alamitos, CA, USA, May 12-15, 2003.

[5] E. Deelman, J. Blythe, Y. Gil, and C. Kesselman. Workflow Management in GriPhyN. *The Grid Resource Management*, Kluwer, Netherlands, 2003.

[6] E. Deelman et al. Mapping Abstract Complex Workflows onto Grid Environments. *Journal of Grid Computing*, 1:25-39, Kluwer Academic Publishers, Netherlands, 2003.

[7] T. Fahringer et al. Truong. ASKALON: a tool set for cluster and Grid computing. *Concurrency and Computation: Practice and Experience*, 17:143-169, Wiley InterScience, 2005.

[8] D. Fernández-Baca. Allocating Modules to Processors in a Distributed System. *IEEE Transactions on Software Engineering*, 15(11): 1427-1436, November 1989.

[9] I. Foster and C. Kesselman (editors), *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann Publishers, USA, 1999.

[10] A. Geppert, M. Kradolfer, and D. Tombros. Market-based Workflow Management. *International Journal of Cooperative Information Systems,* World Scientific Publishing Co., NJ, USA, 1998.

[11] V. Hamscher et al. Evaluation of Job-Scheduling Strategies for Grid Computing. In *1st IEEE/ACM International Workshop on Grid Computing (Grid 2000),* Springer-Verlag, Heidelberg, Germany, 2000; 191-202.

[12] S. Hwang and C. Kesselman. Grid Workflow: A Flexible Failure Handling Framework for the Grid. In *12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*, Seattle, Washington, USA, IEEE CS Press, Los Alamitos, CA, USA, June 22 - 24, 2003.

[13] G. von Laszewski. Java CoG Kit Workflow Concepts for Scientific Experiments. *Technical Report*, Argonne National Laboratory, Argonne, IL, USA, 2005.

[14] B. Ludäscher et al. Scientific Workflow Management and the KEPLER System. *Concurrency and Computation: Practice & Experience*, Special Issue on Scientific Workflows, to appear, 2005

[15] A. Mayer et al. Workflow Expression: Comparison of Spatial and Temporal Approaches. In *Workflow in Grid Systems Workshop*, GGF-10, Berlin, March 9, 2004.

[16] S. McGough et al. Workflow Enactment in ICENI. In *UK e-Science All Hands Meeting*, Nottingham, UK, IOP Publishing Ltd, Bristol, UK, Sep. 2004; 894-900.

[17] T. Oinn et al. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045-3054, Oxford University Press, London, UK, 2004.

[18] S. S. Song, Y. K. Kwok, and K. Hwang. Trusted Job Scheduling in Open computational Grids: Security-Driven heuristics and A Fast Genetic Algorithm. In *19th IEEE International Parallel & Distributed Processing Symposium (IPDPS-2005)*, Denver, CO, USA., IEEE CS Press, Los Alamitos, CA, USA., April 4-8, 2005.

[19] T. Tannenbaum, D. Wright, K. Miller, and M. Livny. Condor - A Distributed Job Scheduler. *Beowulf Cluster Computing with Linux*, The MIT Press, MA, USA, 2002.

[20] I. Taylor, M. Shields, and I. Wang. Resource Management of Triana P2P Services. *Grid Resource Management*, Kluwer, Netherlands, June 2003.

[21] W3C. Extensible Markup Language (XML) 1.0

[22] J. Yu and R. Buyya. A Novel Architecture for Realizing Grid Workflow using Tuple Spaces. In *5th IEEE/ACM International Workshop on Grid Computing (GRID 2004)*, Pittsburgh, USA, IEEE CS Press, Los Alamitos, CA, USA, Nov. 8, 2004.

[23] J. Yu and R. 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.

XML Database Support for Distributed Execution of Data-intensive Scientific Workflows*

Shannon Hastings†, Matheus Ribeiro‡, Stephen Langella†, Scott Oster†, Umit Catalyurek†, Tony Pan†, Kun Huang†, Renato Ferreira‡, Joel Saltz†, Tahsin Kurc†

† Dept. of Biomedical Informatics
The Ohio State University
Columbus, OH, 43210

‡ Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, MG - Brazil

Abstract

In this paper we look at the application of XML data management support in scientific data analysis workflows. We describe a software infrastructure that aims to address issues associated with metadata management, data storage and management, and execution of data analysis workflows on distributed storage and compute platforms. This system couples a distributed, filter-stream based dataflow engine with a distributed XML-based data and metadata management system. We present experimental results from a biomedical image analysis use case that involves processing of digitized microscopy images for feature segmentation.

1 Introduction

The main purpose of data collection is to better understand the problem at hand and predict, explain, and extrapolate potential solutions and outcomes. On one hand, advances in computational and data acquisition technologies improved the resolution and speed at which a researcher can collect data. On the other hand, because of the increasing complexity and size of scientific datasets, data analysis is increasingly becoming a major challenge in research. Scientific data analysis involves several tasks, including 1) querying, retrieval, and integration of data of interest from large and distributed datasets, 2) simple and complex operations on data, and 3) inspection and visualization of results. By composing individual tasks into data analysis workflows, a researcher can create and execute several (potentially new) analysis paths efficiently. This can lead to a better insight to the problem, which may not be possible by processing of data by a single task only. A complex scientific workflow can consist of multiple stages of tasks organized into networks of operations and hierarchical structures (i.e., a task may itself be another workflow).

This work examines the effective application of distributed XML database support in scientific data analysis workflows. XML has become a de facto standard for data exchange in Web and Grid environments. A large number of tools have been developed for creating, parsing, validating, and querying XML documents. With an XML-aware approach, it becomes possible for both clients and application developers to leverage these tools. This paper makes the following contributions: 1) We identify a list of functionality that is desired for scientific workflows and can benefit from XML database support. 2) We describe a software framework that implements the desired functionality by coupling a distributed filter-stream based data processing middleware with a distributed XML data management middleware. The salient features of this system include support for managing data types, which are input to and output from a workflow component, as XML schemas, support for management of workflow descriptions, support for distributed execution of workflows, and on-demand database creation to store and retrieve output datasets and intermediate data products. We present an experimental evaluation of the system in an image analysis use case that involves processing of large biomedical images for feature segmentation.

2 Related Work

A number of research projects have developed tools and runtime infrastructure to support composition and execution of scientific workflows. Because of space constraints, we briefly review some of the related work. A good list and survey of workflow systems can be found at http://www.extreme.indiana.edu/swf-survey/.

The Chimera [4] system implements support for estab-

*This research was supported in part by the National Science Foundation under Grants #ACI-9619020 (UC Subcontract #10152408), #EIA-0121177, #ACI-0203846, #ACI-0130437, #ANI-0330612, #ACI-9982087, #CCF-0342615, #CNS-0406386, #CNS-0426241, Lawrence Livermore National Laboratory under Grant #B517095 (UC Subcontract #10184497), NIH NIBIB BISTI #P20EB000591, Ohio Board of Regents BRTTC #BRTT02-0003. Contact Author: Tahsin Kurc, kurc@bmi.osu.edu.

lishing virtual catalogs that can be used to describe and manage information on how a data product in an application has been derived from other data. This information can be queried, and data transformation operations can be executed to regenerate the data product. The Pegasus project develops systems to support mapping and execution of complex workflows in a Grid environment [3]. The Pegasus framework uses the Chimera system for abstract workflow description and Condor DAGMan and schedulers [5] for workflow execution. It allows construction of abstract workflows and mappings from abstract workflows to concrete workflows that are executed in the Grid. The Kepler project [10] develops a scientific workflow management system based on the notion of actors. Application components can be expressed as Actors that interact with each other through channels. The actor-oriented approach allows the application developer to reuse components and compose workflows in a hierarchical model. Adapting Computational Data Streams is a framework that addresses construction and adaptation of computational data streams in an application [8]. A computational object performs filtering and data manipulation, and data streams characterize data flow from data servers or from running simulations to the clients of the application. Dv [1] is a framework based on the notion of *active frames*. An active frame is an application-level mobile object that contains application data, called *frame data*, and a *frame program* that processes the data. Active frames are executed by *active frame servers* running on the machines at the client and at remote sites.

Our work differs from these projects in that we focus on management of metadata associated with workflows (e.g., definition of a workflow), input/output datasets, and data types exchanged between workflow components using a generic, XML-based metadata and data management system. The system presented in this paper allows storage and retrieval of data and workflow definitions as XML documents conforming to well-defined schemas and enables use of common protocols for storing and querying these documents. Our system could be used by other systems in order to store workflow definitions and instance data. Similarly, our system could use, for example, Condor [5] and Pegasus [3] for scheduling of computations in a Grid environment.

Moreau et. al. [11] propose the use of XML schemas to describe the logical structure of a scientific dataset. The mapping of the logical structure to the physical layout of the dataset is done through mapping documents. The goal is to provide a conceptual view of the dataset, with which applications and workflows can interact without worrying about the physical data format. Their approach is similar to ours in that they use XML schemas to define data types. However, our system, specifically the underlying Mobius framework [7, 9], also provides support for coordinated management and versioning of schemas, on-demand XML database creation, database federation, and querying in a distributed environment. Unlike the XDTM system in [11], which creates a mapping from an XML schema to physical format by mapping documents, the current implementation of Mobius manages XML documents in an XML database, called MakoDB [7, 9], layered on a relational database system. For a given XML schema, MakoDB automatically creates database tables and a mapping of the schema to these tables for efficient storage and querying of XML documents conforming to the schema.

3 Desired Functionality

In order to support complex scientific workflows, a workflow system should address a wide range of requirements. These requirements include user interfaces and languages for easy composition of workflows, workflow scheduling and execution, monitoring of workflows, management of datasets, reliability and robustness, and unified access to sources in the environment, to name a few. Most data analysis applications can be expressed as a network of data processing components that exchange data and control information to execute in a coordinated way. In this paper, we focus on data and metadata centric requirements associated with such workflows.

Strongly typed data. In a distributed and collaborative environment, a data analysis workflow may be composed of components developed by multiple, potentially independent researchers. In such an environment, metadata and data definitions play an important role. Metadata for workflows and definitions of data structures (data objects) exchanged among workflow components can provide a mechanism to ensure a workflow is composed correctly. Moreover, having metadata associated with workflows makes it easier to store, manage, and search for instances of workflows. Web and Grid services standards aim to address the related issues of interoperability between services and components. Nevertheless, support for management of data definitions (i.e., the structure of data objects and datasets, which are input to and output from workflow components) promotes data to be a first class citizen. One advantage of this data centric approach is that strongly typed data provides an additional mechanism to check if two interacting components in a workflow are compatible. Another advantage is, even if the data objects produced and consumed by two components are different, a user or application developer can inspect the structure of the data objects and implement transformation components that will map or convert one data object to the other.

On-demand creation and management of distributed databases. A workflow can be executed by having all its components run concurrently and exchange data on the fly across wide- or local-area networks. Another approach

would be to use a storage system to act as a *persistent data channel* between components at different stages of the workflow. Here, we use the term persistent data channel in the sense that a workflow component's output data object that is stored in the system can be used by multiple invocations of another component in the workflow or a component in another workflow. Such an approach allows more flexible scheduling of stages of a workflow onto available resources. If strongly typed data objects are persisted in the environment, they can be shared, searched, and queried. This not only enables persistent data channels, but also allows other clients and programs to interact with those data objects. By examining the intermediate datasets, a collaborator can better understand how the researcher arrived at her results. Furthermore, intermediate datasets could be utilized as input to a new workflow that builds on the original workflow. This could potentially provide a significant runtime performance improvement as redundant workflow steps do not need to be recomputed if they are shared between multiple workflows [12]. In order to support this type of functionality, the system should be able to create a database of data objects on demand and make it available for remote access. The system should take advantage of storage clusters for large storage space and high I/O performance.

Management of data analysis workflows. Researchers should be able to compose, register, and version workflows. The definitions and instances of workflows should be managed in a standard and efficient way so that researchers can share workflows and reference others' workflows in theirs. The system should also allow a researcher to version workflows (e.g., to add new analysis components or modify the order of data processing steps) and manage the updated instances of workflows.

Efficient execution of data analysis workflows. To speed up complex operations on data and parameter studies, the system should support execution of analysis workflows and manage flow of data within the network of components in a heterogeneous and distributed environment. It should also allow check-pointing of intermediate results so that the workflow can be restarted after a failure or the user can carry out interactive inspection on the data at a later time.

4 System Architecture

We have developed a software infrastructure that implements the functionality presented in Section 3 by coupling an XML-based distributed data and metadata management system, Mobius [7], with a distributed dataflow middleware, DataCutter [2].

4.1 Mobius and DataCutter

Mobius is a middleware framework designed for efficient metadata and data management in dynamic, distributed environments. It provides a set of generic services and protocols to support 1) creation, management, and versioning of XML schemas, 2) on-demand creation and federation of databases conforming to the XML schemas managed by the system, and 3) querying of these databases in a distributed environment. Its services employ XML schemas to represent metadata definitions and XML documents to represent and exchange metadata instances.

The *Mobius Global Model Exchange* (GME) service provides a protocol for publishing, versioning, and discovering XML schemas. It is implemented as an architecture similar to Domain Name Server (DNS), in which there are multiple GMEs each of which is an authority for a set of namespaces. A schema is stored in GME under a name and namespace specified by the application developer and is given a version number. We refer to the tuple consisting of the schema's name, its namespace, and its version number as the global name id (GNI) of the schema. The GME provides support for a schema to reference entities already existing in other schemas and in the global schema defined by a researcher. Other services such as storage services can use the GME to match instance data with their data type definitions. The *Mobius Mako* service exposes data sources as XML data services through a set of well defined interfaces and provides support for storing, updating, retrieving, and querying data as XML documents. The runtime support allows user-defined data types (represented as XML schemas) to automatically manifest custom databases at runtime, and data adhering to these data types to be stored in these databases. Any data instance received by a Mako server is validated against an XML schema as retrieved from the GME server and its elements are indexed. A Mako server can be configured to accept only a specific set of XML schemas. Clients and other services may query XML documents within a Mako by sending it an XPath statement.

DataCutter [2] implements a coarse-grained dataflow system using a filter-stream model. A DataCutter application consists of application-specific components, called *filters*, and one or more *filter groups*. Filters exchange data through a *stream* abstraction, which denotes uni-directional flow of data from one filter to another. The application filters read buffers from their input ports, perform application-defined processing on the buffer, and then write it to their output ports. The DataCutter runtime allows combined use of task- and data-parallelism. Filters can be placed on different machines and multiple copies of a filter can be created and executed across heterogeneous collections of storage and compute nodes.

4.2 Implementation

In our system, the input and output data types of a component in a workflow can be described by data schemas. A data schema is an application specific entity and can describe any user-defined structure and attributes that can

be encoded as an XML schema. For workflows, we have adapted the process modeling schema (PMS) developed for the Distributed Process Management System (DPM) [6]. It defines a hierarchical model starting with a *workflow* entity which contains several *jobs*. Each job represents an application which is composed of a set of *components*. The name attribute of a component allows the component to be named such that it can be uniquely identified. Each component entity also contains input and output list attribute. An optional placement attribute specifies how many copies of a component should be run in the environment and which nodes to run the copies of the component. The PMS and data schemas are registered in and managed by the Mobius GME service.

Our system uses the Mako service of Mobius as the backbone for storage and management of data instances. We utilize a collection of Makos distributed across storage nodes to provide a distributed data management service. Workflow instance descriptions, metadata describing the datasets, and data instances are stored in Makos. An instance of the application workflow is modeled by a directed acyclic task graph of components represented by an XML document conforming to the PMS. The instance specifies the function names and locations of individual components, the number of copies and placement of copies for a component, persistent check-pointing locations in the workflow (which tells the execution environment that output from check-pointed components should be stored as intermediate datasets in the system), input and output datasets (conforming to some registered schema), and an optional data selection criteria (which specifies the subset of data elements from input datasets to be processed). Using Mako client APIs, clients can search for a particular workflow and execute it using the distributed execution service.

The distributed execution service builds on DataCutter and is responsible for instantiation of components on distributed platforms, management of data flow between components, and data retrieval from and storage to distributed collections of Makos. We have implemented two filters, *MakoReader* and *MakoWriter*, that provide interface between Mako servers and other filters in the workflow. The *MakoReader* filter retrieves data from Mako servers and passes them to the first stage filters in the workflow. The *MakoWriter* filter can be placed between two application filters, which are connected to each other in the workflow graph, if the output of a filter needs to be check-pointed. The last stage filters in the workflow also connect to *MakoWriter* filters to store output in Mako servers. To maximize I/O parallelism when data is accessed, the system utilizes data distribution techniques for storing data through the MakoWriter filter. Currently, round-robin and demand-driven (based on the data ingestion rate of individual Makos) strategies are implemented for data distribution. The execution of a workflow and check-pointing can be done stage-by-stage (i.e., all the data is processed by one stage and stored on Mako servers before the next stage is executed) or in pipelined fashion (i.e., all stages execute concurrently; the *MakoWriter* filters interspersed between stages both send data to Mako servers and pass it to the next filter in the workflow). In addition to MakoWriter and MakoReader filters, any workflow component can utilize Mako client APIs to store XML documents and can retrieve data using XPath expressions.

5 Experimental Evaluation

We performed an evaluation of our framework using PC clusters and an image analysis use case. The first set of experiments examines the performance of the system as the number of Mako servers that can store data is varied. A cluster with 7 nodes was used for this experiment. Each node of the cluster consists of two AMD Opteron processors with 8 GB of memory, 1.5 TB of disk storage in RAID 5 SATA disk arrays. The nodes are inter-connected via a Gigabit switch. The workflow consists of a simple chain of *MakoReader->Inverter->MakoWriter* filter group with 7500 images as input – each image was a 256x256-pixel gray scale image. The Inverter filter inverts the color of each pixel in an image. In the experiments, the number of Inverter filter copies was fixed at 7 and each copy was executed on one of the nodes. The input images were distributed evenly across an increasing number of Mako servers as the number of MakoReader filters increased. The results of these experiments are shown in Figure 1. The numbers in the graphs are the total execution time for processing 7500 images. The bars labeled as "Reader" and "Writer" show the execution times when the number of MakoReader and MakoWriter filters is varied, respectively. We observe that the execution time decreases as the number of Mako servers is increased. As more Makos are added, better I/O parallelism is achieved.

In the next set of experiments, we use a biomedical image analysis application implemented as a chain of data processing operations, as shown in Figure 2. This application performs segmentation of the labyrinth layer in a digitized microscopy image of a mouse placenta[1]. The implementation consists of two main stages. The RedFinder, Counter, and Histogram Aggregation filters form the first processing stage of the application and process data in filter-stream fashion. The second stage consists of a single filter referred to here as the PCA filter. Data exchange between the two stages is done through the XML database support. The output of the Counter filter is stored in Mobius and retrieved by the PCA filter. We now briefly describe these stages.

[1]The segmented region can be used to carry out quantitative examination of phenotypes and measurements of tissue structure within the placenta in cancer research.

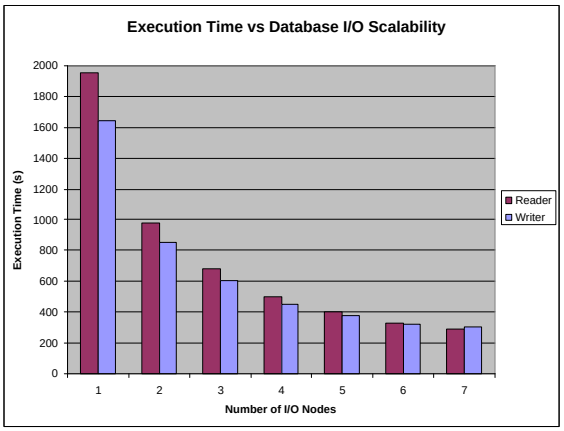


Figure 1. Data I/O Scalability Experiments. "Reader" and "Writer" bars denote the experiments, in which the number of MakoReader and MakoWriter filters is varied, respectively.

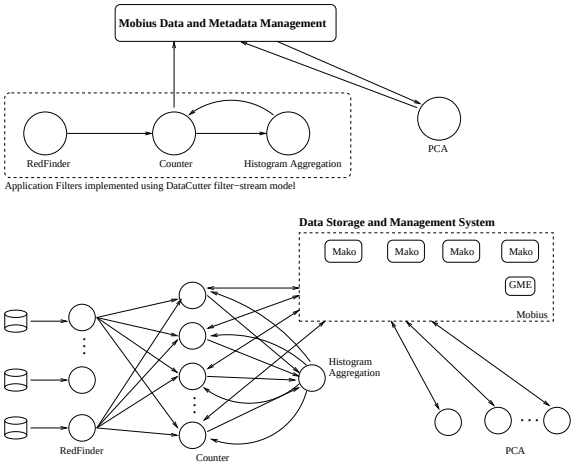


Figure 2. Top: The network of filters which form the processing skeleton of the application. Bottom: An instance of the application with multiple copies of RedFinder, Counter, and PCA filters.

RedFinder, Counter, Histogram Aggregation: These filters find red pixels in regions with high red blood cell (RBC) density. The RedFinder filter reads image chunks, each of which represents a rectangular subregion of the image – in the experiments, input images are stored in files in the system – and determines if a pixel is red or not. The coordinates of red pixels are sent to the Counter filter, which counts the number of neighbors of each red pixel. A histogram of red pixel density is generated by the Histogram Aggregation filter. Using the histogram, the regions with high RBC density (specified by a user-defined threshold value) are determined. The Counter filter finds the red pixels whose coordinates fall into the high density regions, groups them into chunks, creates an XML document for each chunk, and sends using the Mobius client API the XML documents along with the chunk to Makos for storage. The schema for the XML document consists of attributes that specify the encoding type of the image (JPEG, TIFF, raw RGB, etc.), the name (or id) of the image, the id and bounding box of the chunk, and the size of the chunk. The chunk is submitted as an attachment to the XML document.

PCA: This filter retrieves the chunks stored by the Counter filter in the system and processes them to: 1) Compute the principal direction of the high RBC density regions. We apply principal component analysis (PCA) to the high RBC density regions to determine the orientation of these regions. 2) Determine the bounding box for the labyrinth layer by projecting pixels along the principle direction and filtering out those pixels whose projected coordinates fall outside a user-defined range. Multiple copies of the PCA filter can be instantiated to process an image in parallel. During execution, every filter copy first submits an XPath query, which specifies the id of the image, to Mobius and retrieves the list of chunks that satisfy the query. This list is partitioned evenly among filter copies in round-robin fashion. Each filter copy then retrieves the chunks assigned to itself from Mobius for processing using XPath queries. The output from the second stage conforms to the same schema that defines the output of the Counter filter and is stored in Mobius.

We carried out an experimental evaluation of the application implementation on a PC cluster, referred to here as *OSUMED*. Each node has a Pentium III 900MHz CPU, 512MB main memory, and three local disks (300GB local storage space). The nodes in the cluster are inter-connected through a 100Mbit/sec Ethernet Switch. We used digitized microscopy images, each of which was 40GB in size. The results represent the execution time for a single image averaged over 3 images. The amount of data stored in Mobius by the Counter and PCA filters was equal to 1.4GB and 800MB per image, respectively.

In Figure 3, the execution time of the Counter filter is broken down into the operations performed by that filter; doing neighborhood computations and writing data to Mobius. In these experiments, we executed four Mako instances. The round-robin distribution strategy was employed when inserting data to backend Mako instances. As is seen from the figure, the overhead of storing data in Mobius is very small compared to the neighborhood computations. The overhead of neighborhood computations decreases as the number of filter copies is increased as expected. The cost of storage in Mobius remains almost constant as the number of backend Makos was fixed at 4 in these experiments. Our results show that interaction with the XML-based storage services do not become a bottleneck in this configuration as the number of filters writing data is increased. The last set of experiments shows the execution

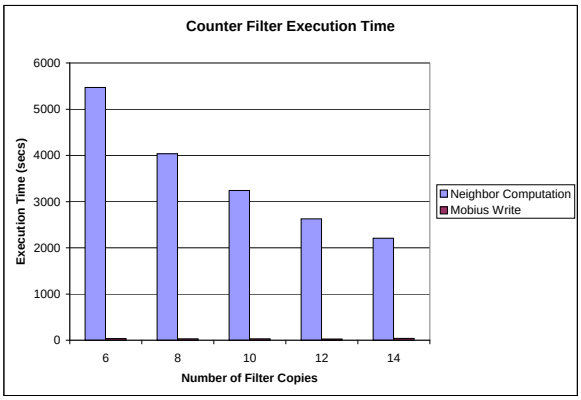


Figure 3. The breakdown of execution of the Counter filter.

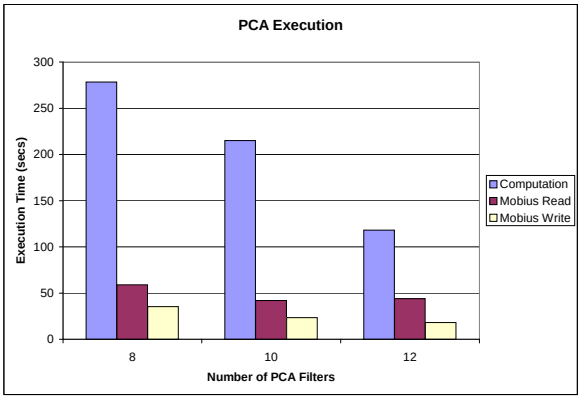


Figure 4. The execution time of the PCA filter as the number of filter copies is varied. The figure also shows the time spent reading data from Mobius and writing data to Mobius.

time of the second stage of the image analysis application (the PCA computations). In our implementation, the second stage is executed after the first stage is completed, i.e., after the redFinder-Counter-Histogram Aggregation filter group has finished its execution. As is seen from Figure 4, the PCA execution time decreases as more filters are executed. The I/O overhead (Mobius Read/Write) is less than the PCA execution time for this experimental setup as well. However, since the number of Mobius Mako servers is fixed, the cost of I/O remains almost constant.

6 Conclusions

This paper examined how XML database support can be applied in the context of scientific workflows. We argue that XML is a viable technology for management and execution of workflows. To demonstrate our ideas, we presented and evaluated a system that uses a distributed XML-based metadata/data management system in tandem with a component-based distributed execution engine.

References

[1] M. Aeschlimann, P. Dinda, J. Lopez, B. Lowekamp, L. Kallivokas, and D. O'Hallaron. Preliminary report on the design of a framework for distributed visualization. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, pages 1833–1839, Las Vegas, NV, June 1999.

[2] M. D. Beynon, T. Kurc, U. Catalyurek, C. Chang, A. Sussman, and J. Saltz. Distributed processing of very large datasets with DataCutter. *Parallel Computing*, 27(11):1457–1478, Oct. 2001.

[3] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, and S. Koranda. Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing*, 1(1), 2003.

[4] I. Foster, J. Voeckler, M. Wilde, and Y. Zhao. Chimera: A virtual data system for representing, querying, and automating data derivation. In *Proceedings of the 14th Conference on Scientific and Statistical Database Management*, 2002.

[5] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke. Condor-G: A computation management agent for multi-institutional grids. In *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*. IEEE Press, Aug 2001.

[6] S. Hastings. Distributed architectures: A java-based process management system. Master's thesis, Computer Science Department, Rensselear Polytechnic Institute, 2002.

[7] S. Hastings, S. Langella, S. Oster, and J. Saltz. Distributed data management and integration: The mobius project. In *GGF Semantic Grid Workshop 2004*, pages 20–38. GGF, June 2004.

[8] C. Isert and K. Schwan. ACDS: Adapting computational data streams for high performance. In *14th International Parallel & Distributed Processing Symposium (IPDPS 2000)*, pages 641–646, Cancun, Mexico, May 2000.

[9] S. Langella, S. Hastings, S. Oster, T. Kurc, U. Catalyurek, and J. Saltz. A distributed data management middleware for data-driven application systems. In *Proceedings of 2004 IEEE International Conference on Cluster Computing*, September 2004.

[10] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows, to appear*, 2005.

[11] L. Moreau, Y. Zhao, I. Foster, J. Voeckler, and M. Wilde. XDTM: the XML Dataset Typing and Mapping for Specifying Datasets. In *Proceedings of the 2005 European Grid Conference (EGC'05)*, Amsterdam, Netherlands, Feb. 2005.

[12] D. Thain, J. Bent, A. Arpaci-Dusseau, R. Arpaci-Dusseau, and M. Livny. Pipeline and batch sharing in grid workloads. In *Proceedings of High-Performance Distributed Computing (HPDC-12)*, pages 152–161, Seattle, Washington, June 2003.

Scheduling of Scientific Workflows in the ASKALON Grid Environment*

Marek Wieczorek, Radu Prodan and Thomas Fahringer
Institute of Computer Science, University of Innsbruck
Technikerstraße 21a, A-6020 Innsbruck, Austria
marek@dps.uibk.ac.at

ABSTRACT

Scheduling is a key concern for the execution of performance-driven Grid applications. In this paper we comparatively examine different existing approaches for scheduling of scientific workflow applications in a Grid environment. We evaluate three algorithms namely genetic, HEFT, and simple "myopic" and compare incremental workflow partitioning against the full-graph scheduling strategy. We demonstrate experiments using real-world scientific applications covering both balanced (symmetric) and unbalanced (asymmetric) workflows. Our results demonstrate that full-graph scheduling with the HEFT algorithm performs best compared to the other strategies examined in this paper.

1. INTRODUCTION

Scheduling of scientific workflow applications on the Grid is a challenging problem, which is an ongoing research effort followed by many groups. Deelman [9] distinguishes several workflow processing strategies covering trade-offs between dynamicity and look-ahead range in workflow processing. In [3] Deelman proposed a scheduling strategy based on initial partitioning of the workflow into sequential sub-workflows, that are scheduled sequentially one after another. Prodan [10] applied genetic algorithms [7] to schedule the whole workflow at once, and rescheduling it many times during the execution. These approaches were not compared against each other.

In this paper we examine three scheduling algorithms to evaluate their performance for scheduling scientific workflows in Grid environments. The scheduling algorithms comprise a genetic algorithm similar to the one presented in [10], the well-known HEFT algorithm [15], and a "myopic" algorithm. The HEFT algorithm is an extension for heterogeneous environments of the classical list scheduling algorithm [8]. HEFT is a simple and computationally inexpensive algorithm, which schedules workflows by creating an ordered list of tasks out of the workflow, and mapping the tasks to the resources in the most appropriate way. Execution order is based on the list created in the first two phases of the algorithm. The last algorithm we applied is a simple "myopic" algorithm, similar to the Condor DAGMan [12] resource broker, which schedules the next task onto the best machine available without any long-term optimization strategy. The Grid model applied by us in the experiments assumes high availability rate and good control over the resources by the scheduler. This is not always assumed by many other Grid research groups, but it is usually the case for scientific workflows executed in research institutions.

Additionally, we compared different scheduling strategies including full graph scheduling and incremental workflow partitioning strategy [3]. The Myopic algorithm can be considered as a just-in-time scheduling strategy, as the scheduling decisions made by the algorithm are optimized for the current time instance.

In the remainder of this paper, we evaluate the scheduling approaches through a series of experiments. We show, that the HEFT algorithm is more effective and less time consuming than the genetic algorithms applied in [10]. HEFT also performs substantially better than a simple Myopic algorithm. We also show that the workflow partitioning approach described in [3] does not appear to imply any advantage over full-graph scheduling. For the class of strongly unbalanced (asymmetric) workflows we highlight poor performance of the incremental workflow partitioning and simple scheduling algorithms. Full-ahead scheduling with the HEFT algorithm appears to perform best for unbalanced workflows.

2. ASKALON ENVIRONMENT

ASKALON [5] is a Grid environment for composition and execution of scientific workflow applications. The workflow model adopted in ASKALON is described in Section 3. The scheduler optimizes for performance using the execution time as the most important goal function. The scheduler interacts with the enactment engine (see Fig. 1) which is a service that supervises the reliable and fault tolerant execution of the tasks and transfer of the files. The resource

*This research is partially supported by the Austrian Science Fund as part of the Aurora project under contract SFBF1104 and the Austrian Federal Ministry for Education, Science and Culture as part of the Austrian Grid project under contract GZ 4003/2-VI/4c/2004.

broker and the performance predictor are auxiliary services which provide information about the resources available on the Grid, and predictions about expected execution times and data transfer times. The performance monitoring service provides up-to-date status information of the application execution and of the Grid environment. This information can be used by the scheduler to make a decision about rescheduling.

The scheduler itself consists of several components. The workflow evaluator transforms the dynamic and compact representation of the workflows in a static structure as described in Sec. 3. The scheduling engine performs the actual scheduling, applying one of the alternative scheduling algorithms. The event generator is meant for generation of rescheduling events to cope with the dynamic nature of workflows and the Grid and is currently being implemented.

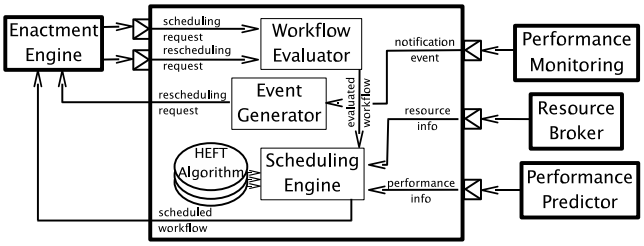


Figure 1: ASKALON environment architecture

3. WORKFLOW MODEL

Scientific workflows executed in the ASKALON environment are based on the model described in the AGWL specification language [13]. AGWL documents can express simple DAGs as well as more sophisticated workflow graphs containing loops and conditional branches which impose control flow decisions that can only be decided at runtime. The condition of a conditional branch (either *if-then* or *switch*) may be evaluated in various ways for different executions, or a *while-loop* may have different number of iterations. Furthermore, *parallel-for* loops are introduced to specify a large number of parallel activities (hundreds) in a compact form, for scalability reasons. Such parallel-for constructs may be evaluated differently at run-time, depending on the parameters of the current execution. The actual number of parallel activities specified by a parallel-for construct may not be known at the beginning of the workflow execution. In order to apply a full-graph scheduling algorithm, all such uncertainties have to be resolved. To this end, we make assumptions about the actual evaluation of the control structures. If an assumption fails, the scheduler transforms the workflow once again in the proper way and reschedules it. This approach may bring considerable benefit if the structure of the workflow is predicted correctly (especially, when a strong unbalance in the workflow is detected). If the conditions are predicted incorrectly, the workflow execution time is the same as in the case of a just-in-time strategy which schedules only those parts of the workflow that are resolved at the moment of scheduling. Fig. 5-8 show two such workflow transformations applied to real Grid workflow applications (see Section 6).

4. SCHEDULING ALGORITHMS

The scheduling algorithms under consideration map tasks as part of workflows onto Grid sites (clusters). Each Grid site consists of a set of CPUs, each of which is considered as a single computational resource. If a task is executed on a CPU, no other tasks can use the same CPU at the same time. Execution times of tasks and data transfers generated by the performance predictor are given as input data to the scheduler.

4.1 HEFT algorithm

The HEFT algorithm that we applied consists of 3 phases:

1. *Weighting* assigns the weights to the nodes and edges in the workflow;
2. *Ranking* creates a sorted list of tasks, organized in the order how they should be executed;
3. *Mapping* assigns the tasks to the resources.

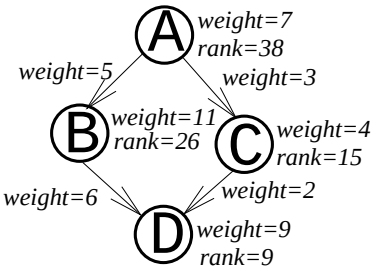
The weights assigned to the nodes are calculated based on the predicted execution times of the tasks. The weights assigned to the edges are calculated based on predicted times of the data transferred between the resources. In homogeneous environments the weights are equal to the predicted times. In heterogeneous environments, the weights must be approximated considering different predictions for execution times on different resources, and for different data transfer times on different data links. Several approximation methods were proposed and compared [11]. Each of them provides different accuracy for different cases. We chose the arithmetic average.

The ranking phase is performed traversing the workflow graph upwards, and assigning a rank value to each of the tasks. Rank value is equal to the weight of the node plus the execution time of the successors. The successor execution time is estimated, for every edge being immediate successors of the node, adding its weight to the rank value of the successive node, and choosing the maximum of the summations. A list of resources is arranged, according to the decreasing rank values. An example workflow graph with the calculated weights and ranks is shown in Fig. 2. The example considers 3 heterogeneous resources R1, R2 and R3. Data transfer is assumed to be equal in both directions between any two of those resources.

In the mapping phase, consecutive tasks from the ranking list are mapped to the resources. For each task, the resource which provides the earliest expected time to finish execution is chosen. The pseudocode of the HEFT algorithm is depicted in Alg. 1.

4.2 Genetic Algorithms

Genetic Algorithms are a part of evolutionary computing, inspired by Darwin's theory of evolution. They represent powerful optimization heuristics, used to search global minima in multi-dimensional search spaces. The basis of the algorithm is to encode possible solutions of the problem into a *population* of *chromosomes*, and subsequently to transform the population using standard operations of *selection*, *crossover* and *mutation*, producing successive *generations*. The selection is driven by an established *fitness function*, which evaluates the chromosomes in terms of accuracy of the represented solutions. Crossover and mutation respond



	R1	R2	R3	*avg*
A	5	8	8	7
B	9	13	11	11
C	3	4	5	4
D	7	10	10	9

execution times on different resources

	R1->R2	R1->R3	R2->R3	*avg*
A->B	6	4	5	5
A->C	4	2	3	3
B->D	7	4	7	6
C->D	1	1	4	2

data transfer time between different resources

Figure 2: Weights and ranks calculated with HEFT algorithm

algorithm 1 HEFT algorithm

T - set of all tasks in the workflow,
E - set of all dependencies in the workflow,
R - set of all available resources,
(t_1, t_2) - dependence between tasks t_1 and t_2
$time(t, r)$ - execution time of task t on resource r,
$time(e, r1, r2)$ - data transfer time of data between resources $r1$ and $r2$ (dependence e in the workflow),
Weighting phase##
for each $t \in T$ do
~ $w(t) = \frac{\sum_{r \in R} time(t,r)}{\#R}$
for each $e \in E$ do
~ $w(e) = \frac{\sum_{r_1, r_2 \in R, r_1 \neq r_2} time(e, r_1, r_2)}{\#R \cdot (\#R - 1)}$
##Ranking phase##
$Succ = \{(t_1, t_2) : t_1, t_2 \in T \wedge (t_1, t_2) \in E\}$
$NSucc = Succ$
$NT = T$
while $NT \neq \{\}$ do
~ $Last = \{t : t \in NT \wedge \neg \exists t_1 : (t, t_1) \in NSucc\}$
~ for each $t \in Last$ do
~ $LS(t) = \{t_1 : (t, t_1) \in Succ\}$
~ $rank(t) = w(t) + max(\{0\} \cup \{r : r = w(t_1) + w((t, t_1)) \wedge t_1 \in LS(t)\})$
~ $NSucc = NSucc \setminus \{(t_1, t) : (t_1, t) \in NSucc\}$
~ end
~ $NT = NT \setminus Last$
end
$ranking\_list = sort(T, rank)$
##Mapping phase##
for $i = \#ranking\_list$ downto 1 do
$t = ranking\_list[i]$
~ Find resource $r \in R$: $finish\_time(t, r)$ is min;
~ Schedule t to r;
~ Mark r as reserved until $finish\_time(t, r)$;
end

to standard biological operations of mutual exchange of a part of body within a pair of chromosomes, and of change of some elements (so-called *genes*) in the chromosomes randomly selected from the population. The end condition of a genetic algorithm is usually the *convergence criterion* which checks how much the best individual found changes between subsequent generations. A maximum number of generations can also be established. The pseudocode of a genetic algorithm is presented in Alg. 2.

algorithm 2 Genetic algorithm

Create the initial population of chromosomes;
while convergence criteria is false do
~ Perform crossover and mutation;
~ Calculate fitness values for the population;
~ Create a new population, based on actual fitness values;
end

Genetic Algorithms are a good general purpose heuristic, which is able to find the optimal solution even for complicated multi-dimensional problems. By transforming a broad population of chromosomes in a semi-random manner, the entire search space is traversed and the search does not end up in a local minimum. However, Genetic Algorithms are not equally appropriate for every possible optimization problem. Solutions of the problem must be properly encoded into the chromosomes, which is not always feasible. Prodan in [10] encoded the actual mapping of tasks to the resources without specifying the order of execution of independent tasks (not linked through control and data flow dependencies) that are scheduled on the same CPU. Therefore this execution order cannot be a subject to optimization. Moreover, Genetic Algorithms tend to be computationally extensive.

4.3 Myopic algorithm

To compare the scheduling algorithms described so far, we developed a simple and inexpensive scheduling algorithm, which makes the planning based on locally optimal decisions. The algorithm represents a class of schedulers covering for instance the Condor DAGMan resource broker which employs the matchmaking mechanism [12]. The pseudocode of the algorithm is described in Alg. 3.

algorithm 3 Myopic algorithm

T - set of all tasks in the workflow,
$NT = T$
while $NT \neq \{\}$ do
~ Find task $t \in NT$: $earliest\_starting\_time(t)$ is min;
~ Find resource $r \in R$: $finish\_time(t, r)$ is min;
~ Schedule t to r;
~ Mark r as reserved until $finish\_time(t, r)$;
~ $NT = NT \setminus \{t\}$
end

The Myopic algorithm can produce reasonably accurate results for rather simple workflows given accurate performance predictions. But it does not provide any full-graph analysis and does not consider the order of task execution.

5. SCHEDULING STRATEGIES

Different scheduling strategies [9] can be applied, considering the trade-off between dynamicity and look-ahead range in workflow processing. *Just-in-time* strategy, in [9] referred to as *in-time local scheduling*, consists of mapping the tasks to the resources, always choosing the most appropriate solution for the current step. This approach benefits from using

the most up-to-date performance data, which is important for the Grid, but on the other hand it neglects the graph structure, and the whole workflow may not be scheduled optimally. At the other extreme, we have *full-ahead planning* where the full-graph scheduling is performed at the beginning of execution. In this case, a sophisticated graph scheduling algorithm can be applied, but the dynamism of the Grid is not considered. Intermediate solutions try to reconcile workflow planning with Grid dynamism, and to find an approach which considers both the workflow structure and the Grid behavior. One of the possible solutions is the workflow partitioning applied in the Pegasus system [3]. It consists of an initial partitioning of the workflow into a sequence of subworkflows, which are subsequently scheduled and executed (see Fig. 3).

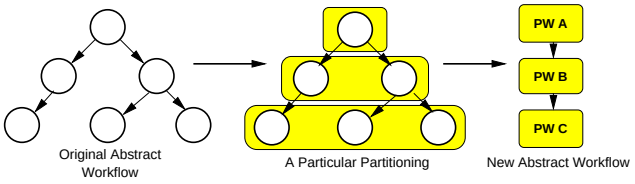


Figure 3: Workflow partitioning in Pegasus [3]

Each partitioning can be characterized by the width of a slice. The width of a slice is expressed as maximal number of node layers within each slice. For instance, the workflow depicted on Fig. 3 was partitioned with one layer per slice (*1-layer partitioning*). Any element of the sequence can be scheduled and executed only if the immediate predecessor has already finished its execution. This approach has an advantage over the simple just-in-time scheduling, as the planner considers more than one task at the time, and has a better overview of the whole graph.

Full-graph scheduling, however, can also be applied in a dynamic way. If we do not consider the initial scheduling as the ultimate decision but only a hint, and if we admit subsequent reschedulings if they are necessary, we can apply a full-graph scheduling algorithm many times during the execution of a workflow. One of the workflows we applied in our experiments belongs to a specific class of strongly unbalanced workflows, which seems to require full-graph analysis for proper scheduling. The workflow contains a parallel section and some of the branches take longer to execute than the others (see Fig. 8). The tasks that belong to the longer branch should, therefore, execute with higher priority than the ones that belong to the shorter branches. One important goal of our experiments was to investigate how the scheduling results depend on the workflow strategy applied for strongly unbalanced workflows.

As our experiments concern scientific workflows executed in research institutions, we assume high availability rate and good control over the resources, what is not always the case for best-effort Grid schedulers. In particular, we assume that the scheduler can have precise information about the resources available in the Grid, and the submissions made by the scheduler are executed as they were requested. We also assume that no failures occur during the execution, so that the execution of the workflow is performed in the same way as it was planned by the scheduler.

6. EXPERIMENTAL RESULTS

In our experiments we compare the HEFT algorithm with a genetic algorithm similar to the one proposed in [10], and with the Myopic algorithm described in Section 4.3. We also compare the full-graph scheduling with the workflow partitioning strategy. As results, we show execution times of the scheduled workflow applications (*execution times*), and the times spent in preparing the schedules (*scheduling times*). The execution times were measured for two scenarios of workflow execution. In the first scenario, we do not provide to the scheduler any performance predictions, so the scheduler has to assume that all the execution times are equal for all tasks on all the resources (scheduling *without performance guidance*). In the second scenario, the scheduler is provided with experience-based performance predictions derived from historical executions (scheduling *with performance guidance*). The predictions were provided to the scheduler in a two-dimensional array, containing the execution time of each task on each computer architecture available in our Grid. The assumption was that each task takes the same execution time on every machine that belongs to the same type (i.e, has the same CPU model, CPU speed and total RAM size).

Experiments were performed incorporating seven Grid sites (clusters) of the Austrian Grid [2] infrastructure with 116 CPUs in total (not all Grid sites were used in all the experiments). In Fig. 4 we present the performance of the individual clusters, where each cluster shows the average execution time of all the workflow tasks executed on a single CPU. As we can see, the fastest cluster is more than three times faster than the slowest one.

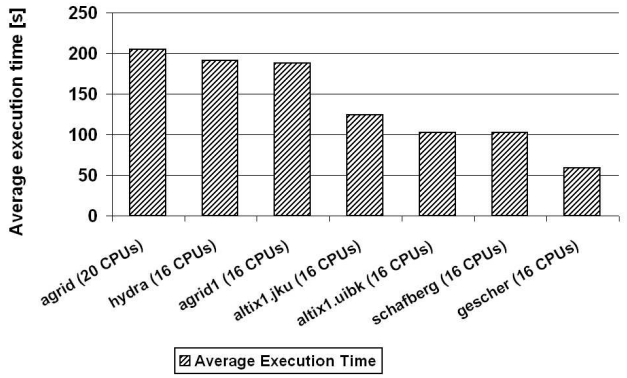


Figure 4: Performance of Grid clusters used in the experiments.

Similarly to execution time predictions, the execution times of the tasks on the sites were measured on the Austrian Grid during a test phase. Time consumed by data transfers between two tasks connected with a data link was considered as constant. We also fixed the middleware overhead introduced by the Globus GSI security [6] and the PBS queuing system [14] to 30 seconds.

We used two real-world workflow applications in our experiments. WIEN2k [1] is a quantum chemistry application developed at Vienna University of Technology. WIEN2k workflow (Fig. 5) is a fully-balanced workflow which contains two parallel sections with possibly many parallel tasks, and an external loop. For our tests we considered the work-

flow with one iteration of the loop, and 250 parallel tasks in each section (Fig. 6). Invmod [4] is a hydrological application developed at the University of Innsbruck, designed for calibration of parameters for the WaSiM tool developed at the Swiss Federal Institute of Technology Zurich. The Invmod workflow (Fig. 7) consists of an outermost parallel loop with many iterations (executed as separate threads), which contains a nested optimization loop. We used this workflow to simulate the common case of strongly unbalanced workflow. If the loops in individual workflow threads have different numbers of iterations (and the iteration numbers are predicted correctly), then the threads may differ significantly with regard to their expected execution times. The workflow used for these experiments (Fig. 8) contains 100 parallel iterations one of which contains 20 iterations of the optimization loop. The remaining 99 iterations contain 10 optimization iterations each. It means, that one of the threads takes approximately twice as much execution time as all others.

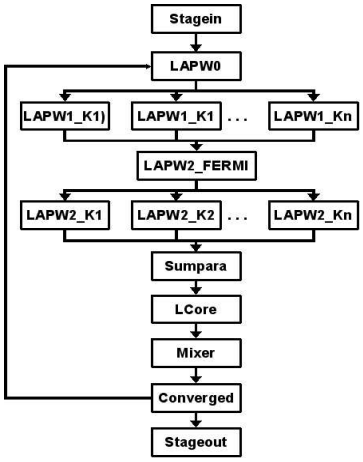


Figure 5: WIEN2k, original workflow.

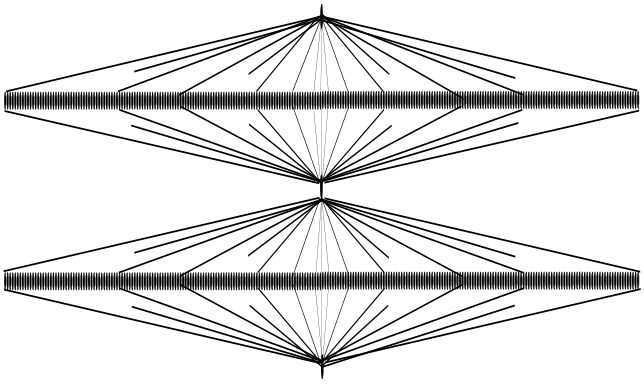


Figure 6: WIEN2k, transformed workflow.

The genetic algorithm that we applied is based on the population of 100 chromosomes transformed in 20 generations, which is not a large number but it allowed us to achieve a good convergence rate with relatively small scheduling time. Probability of crossover was fixed by us to 0.25, and mutation rate to 0.01. We performed workflow partitioning by dividing the workflow into slices with well-defined width (see

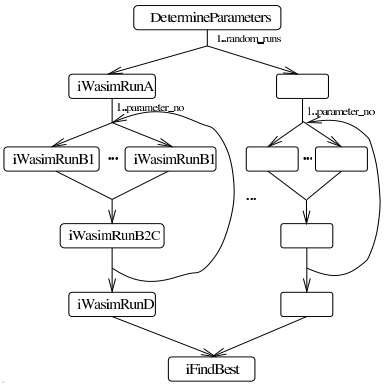


Figure 7: Invmod, original workflow.

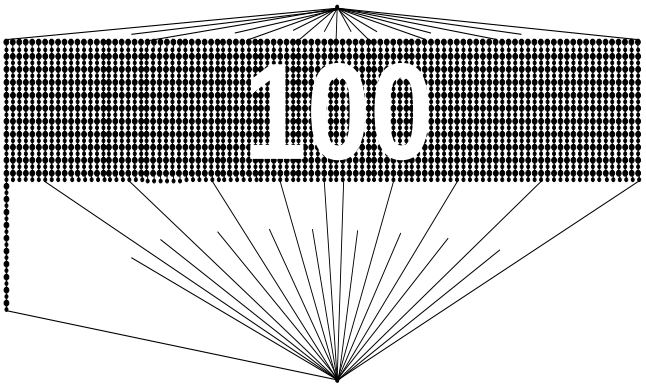


Figure 8: Invmod, transformed workflow.

Section 5). For the WIEN2k workflow (consisting of five layers) we applied a three-layer partioning, and for Invmod workflow (which consists of 44 layers) we applied three different partitionings, with 10, 20 and 30 layers.

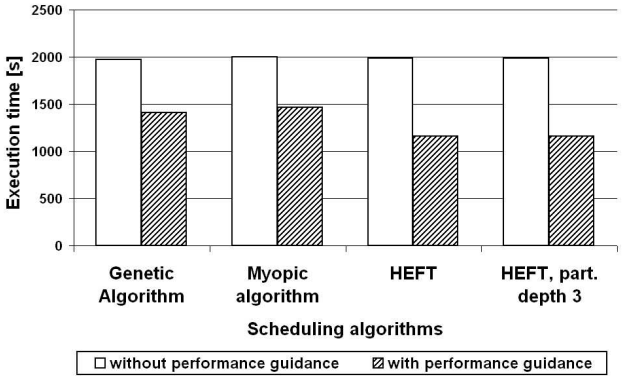


Figure 9: WIEN2k executed in heterogeneous environment, execution time.

The first conclusion we draw from the results (Fig. 9-12) is that performance prediction is very important in heterogeneous Grid environments. For both workflows, the results achieved with performance guidance are in the best case nearly two times better that the results achieved without performance guidance. Performance estimates are clearly important even if they are not highly accurate.

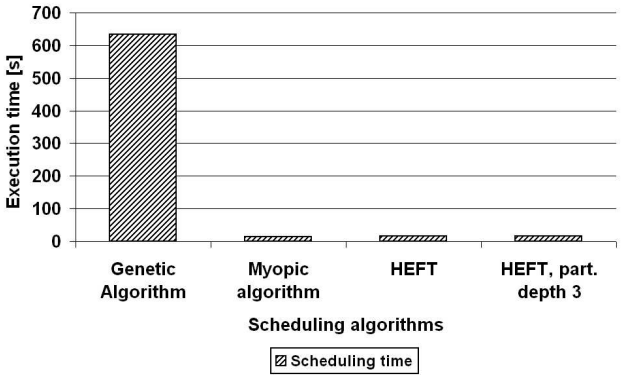


Figure 10: WIEN2k executed in heterogeneous environment, scheduling time.

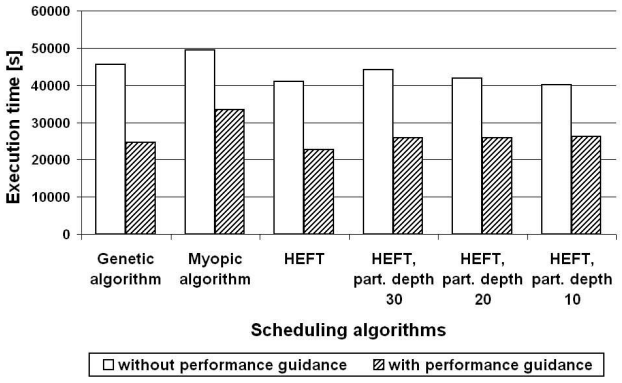


Figure 11: Invmod executed in heterogeneous environment, execution time.

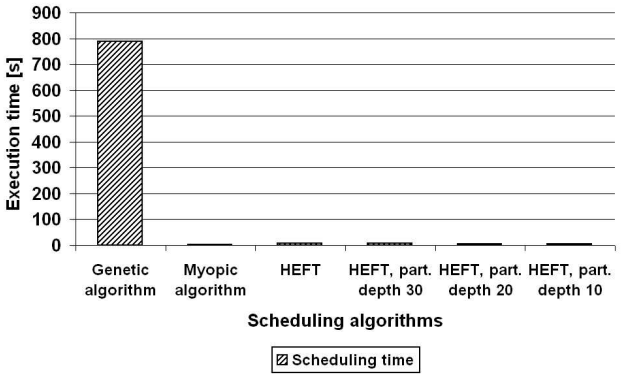


Figure 12: Invmod executed in heterogeneous environment, scheduling time.

Comparing the results measured for the WIEN2k workflow we can notice that HEFT produces much better results than the other algorithms. Execution time of the workflow is 17% shorter than for the genetic algorithm, and even 21% than for the Myopic. The simple solution applied in Myopic appears to be insufficient for large and complicated workflows, and the algorithm produces the worst results. Also the genetic algorithm appears to be not a good method to deal with our problem. It was able to approximate the global maximum, but it did not find the actual best value which lies probably in a "long and narrow corner" of the search space. For the scheduling without performance guidance, where the search space has more regular borders, the genetic algorithm behaves equally good (or even better) than all the other algorithms. Comparing the scheduling times of individual algorithms we can see that the genetic algorithm executes two to three orders of magnitude longer than the others. It means, that even to generate a single population takes much longer than the HEFT algorithm.

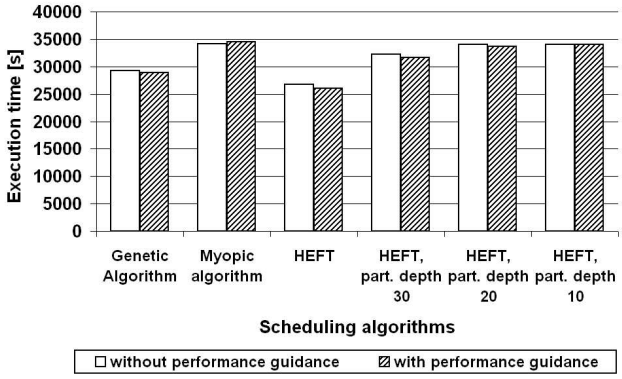


Figure 13: Invmod executed in homogeneous environment, execution time.

The results measured for the Invmod workflow present how individual algorithms deal with strongly unbalanced workflows. As expected, the Myopic algorithm provides the worst results of all, approximately 32% worse than HEFT. The genetic algorithm produces quite good results. It was able to locate the area where the global minimum is located, but it was not able to find the best possible solution, since the order of execution (of independent tasks scheduled to the same CPU) was not considered for optimization. In the workflows scheduled without an established task order, the tasks are executed in an arbitrary order chosen by the runtime system. For a strongly unbalanced workflow, however, the tasks that execute in iterations of the parallel loop with longer execution time should be executed more often than the others, which cannot be done by the runtime system which does not consider global graph structure. Scheduling strategies based on the workflow partitioning were also not able to find the optimal solution, although their results are still better than the one found by the Myopic algorithm. Only the full-graph analysis could find a well performing scheduling solution for imbalanced workflows. Since all of the algorithms (except for the genetic algorithm) execute really fast (less than 20 seconds for large and complicated workflows), there is no reason to apply the partitioning strategy in place of the full-graph analysis.

Fig. 13 presents the execution results of the Invmod workflow on a homogeneous environment (three nearly identical Grid sites). As expected, there is now almost no difference between the scheduling with and without performance guidance, as the execution on each cluster takes the same time. Again, HEFT produces the best results, 24% better than Myopic.

7. CONCLUSIONS AND FUTURE WORK

Scheduling applications on the Grid is of paramount importance to optimize non-functional parameters such as execution time. In this paper we compared three different algorithms examining aspects such as incremental versus full-graph scheduling for balanced versus unbalanced workflows.

Based on two real world Grid workflows we observed that the HEFT algorithm appears to be a good and computationally inexpensive scheduling algorithm that performs better than the other 2 candidates discussed in this paper.

We also investigated a specific class of strongly unbalanced workflows. We demonstrated that any just-in-time scheduling strategy is likely to produce poor results for workflows of this class. Also the workflow partitioning strategy used in Pegasus system [3] appears to have no advantage over the full-graph scheduling, and may produce less efficient results for unbalanced workflows.

We implemented the HEFT algorithm in the ASKALON environment for scheduling scientific workflow applications on the Grid. Future work on the presented scheduling strategy will consist of making it more efficient for heterogeneous environments. We will also examine how typical network scenarios may disrupt the simple scheduling model based on fixed values provided as performance predictions.

8. REFERENCES

[1] P. Blaha, K. Schwarz, G. Madsen, D. Kvasnicka, and J. Luitz. *WIEN2k: An Augmented Plane Wave plus Local Orbitals Program for Calculating Crystal Properties*. Institute of Physical and Theoretical Chemistry, Vienna University of Technology, 2001.

[2] The Austrian Grid Consortium. http://www.austriangrid.at.

[3] Ewa Deelman, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, and Miron Livny. Pegasus: Mapping scientific workflows onto the grid. In *European Across Grids Conference*, pages 11–20, 2004.

[4] Peter Rutschmann Dieter Theiner. An inverse modelling approach for the estimation of hydrological model parameters. In *Journal of Hydroinformatics*, 2005.

[5] Rubing Duan, Thomas Fahringer, Radu Prodan, Jun Qin, Alex Villazon, and Marek Wieczorek. Real World Workflow Applications in the Askalon Grid Environment. In *European Grid Conference (EGC 2005)*, Lecture Notes in Computer Science. Springer Verlag, February 2005.

[6] GT 4.0 Security website. http://www.globus.org/toolkit/docs/4.0/security/.

[7] David E. Goldberg. *Genetic Algorithms in Search, Optimization 6 Machine Learning*. Reading. Addison-Wesley, Massachusetts, 1989.

[8] R. L. Graham. Bounds for certain multiprocessing anomalies. In *Bell System Technical Journal 45*, pages 1563–1581, 1969.

[9] Jarek Nabrzyski, Jennifer M. Schopf, and Jan Weglarz. *Grid Resource Management, State of the Art and Future Trends*. Kluwer, 2003.

[10] Radu Prodan and Thomas Fahringer. Dynamic Scheduling of Scientific Workflow Applications on the Grid using a Modular Optimisation Tool: A Case Study. In *20th Symposion of Applied Computing (SAC 2005)*, Santa Fe, New Mexico, USA, March 2005. ACM Press.

[11] Rizos Sakellariou and Henan Zhao. A hybrid heuristic for dag scheduling on heterogeneous systems. In *IPDPS*, 2004.

[12] The Condor Team. Dagman (directed acyclic graph manager). http://www.cs.wisc.edu/condor/dagman/.

[13] Jun Qin Thomas Fahringer and Stefan Hainzer. Specification of Grid Workflow Applications with AGWL: An Abstract Grid Workflow Language. In *Proceedings of IEEE International Symposium on Cluster Computing and the Grid 2005 (CCGrid 2005)*, Cardiff, UK, May 9-12 2005. IEEE Computer Society Press.

[14] Veridian Systems. PBS: The Portable Batch System. http://www.openpbs.org.

[15] Henan Zhao and Rizos Sakellariou. An experimental investigation into the rank function of the heterogeneous earliest finish time scheduling algorithm. In *Euro-Par*, pages 189–194, 2003.

Efficient calendar based temporal association rule

Keshri Verma , O. P. Vyas
School of Studies in Computer Science
Pt. Ravishankar Shukla University Raipur Chhattisgarh
{keshriverma,opvyas}@rediffmail.com

Abstract

Associationship is an important component of data mining. In real world data the knowledge used for mining rule is almost time varying. The item have the dynamic characteristic in terms of transaction , which have seasonal selling rate and it hold time-based associationship with another item. It is also important that in database, some items which are infrequent in whole dataset but those may be frequent in a particular time period. If these items are ignored then associationship WVW200R3100221-398 result will no longer be accurate. To restrict the time based associationship calendar based pattern can be used [YPXS03]. A calendar unit such as months and days, clock units, such as hours and seconds & specialized units , such as business days and academic years, play a major role in a wide range of information system applications[BX00].

Most of the popular associationship rule mining methods are having performance bottleneck for database with different characteristics. Some of the methods are efficient for sparse dataset where as some are good for a dense dataset. Our focus is to find effective time sensitive algorithm using H-struct called temporal H-mine, which takes the advantage of this data structure and dynamically adjusts links in the mining process [PHNTY01]. It is faster in traversing & advantage of precisely predictable spaces overhead. It can be scaled up to large database by database partitioning, end when dataset becomes dense, conditionally temporal FP-tree. can be constructed dynamically as part of mining.

1. Introduction –

The Associationship is an important component of data mining. It indicates the co-relationship of one item with another. For example Egg ==> coffee (support 3%, confidence 80%) means that 3% of all transaction contain both egg & coffee, and 80% of transaction that have egg also have coffee in them. In real dataset time is one of the important factors. For example egg and coffee may be ordered together primarily between 7 to 11 AM in this interval the support & confidence is 40% but at another interval, support is as low .005% in other transaction [YPXS03]. This discussion suggests that different association rules may be discovered while considering different time intervals associated to it. Many items are introduced or removed form the database, that is items lifespan[ZMTW02] which means that item is valid on specific time interval. To discover such temporal intervals (with calendar information) together with the association rules that hold during the time interval may lead to useful knowledge. If calendar schema is applied in association it is called calendar based temporal association rule.

A hierarchy of calendar concepts determines a calendar schema. A calendar unit such as months and days, clock units, such as hours and seconds & specialized units, such as business days and academic years, play a major role in a wide range of information system applications [BX00]. A calendar schema defines a set of simple calendar – based patterns. Each calendar pattern defines a set of time intervals.

Recent researches in the field of temporal association rule mining are using Apriori based approach. Nevertheless, these proposed approaches may still encounters some difficulties for different datasets such as Sparse or dense dataset. The limitations in these approaches are two fold.

First, huge space is required to perform the mining in Apriori based temporal association rule [JG00]. It generates a huge number of candidates in case of a dataset, which is large and/or sparse. Our first part of the algorithm which if the database is huge and sparse, the temporal approach of FP –tree outperform over Apriori and the space requirement for recursion is a challenge. Thus necessitates improvement in the existing approach. Second part of the algorithm generates Fp-tree when the data have the dense characteristic, no algorithm can bits the performance of FP-tree.

Second, the mining approach should ideally have more scalability. Many existing methods are effective when the dataset are not large. The existing Apriori based temporal

association rule , may easily cause thrashing when dataset become large and sparse.

The approach of frequent pattern mining is to find the complete set of frequent patterns in a given transaction database with respect to a given support threshold. Our data-mining problem is to discover all temporal association rules w.r.t. Calendar schema from a set of time stamped transactions. This paper ,improves an existing frequent pattern tree approach to discover temporal association rule to increase the memory performance over existing one[PHNTY01] , it uses the data structure H-struct, and incorporating temporal aspects with the following progress:

First, a memory based efficient pattern growth algorithm, Temporal H-mine is proposed for mining time based frequent patterns for dataset that can fit in memory. H-struct is used for fast mining on time-based dataset. It has polynomial space complexity and is thus more space efficient than pattern growth method like FP-growth and Tree-Projection when mining sparse dataset, and more efficient than Apriori based frequent pattern mining[JG00].

Second based on H-mine data structure we propose Temporal based H-mine association rule mining algorithm.

Third for dense datasets. H-mine is integrated with Temporal FP-growth dynamically by detecting the swapping condition and constructing FP-tree for effective mining.

The Temporal base H-mine algorithm is scalable in both large and medium size dataset and in both the cases dense and sparse dataset.

The rest of the paper is organized in five section In Section 2, we discuss some related works. In section 3 we define temporal association rule in term of calendar schema. In Section 4 elaborate the extended algorithm of frequent pattern approach ,section 5 shows conclusion & future works and section 6 provides application of above investigation.

2. Related work –

The concept of association rule was introduced as Apriori algorithm [AS94]. Its performance was improved by deploying frequent-pattern growth approach [PH02]. In paper [ORS98] the omission of the time dimension in association rule was very clearly mentioned. Fp-growth algorithm is best if data is dense, & Apriori algorithm performs better if data is sparse, H-mine [PHNTY01] algorithm which is used to mine the dataset from both the cases sparse & dense. A temporal aspect of association rule was given by Juan [JG00]. According to this transaction in the database are time stamped and time interval is specified by the user to divide the data into disjoint segments, like month , days & years. Further The cyclic association rule was introduced by Ozden [ORS98] with minimum support & high confidence. Using the definition of cyclic association rule, It may not have high support & confidence for the entire transactional database. A nice bibliography of temporal data mining can be found in the Roddick literature[RHS00].Rainsford & Roddick presented extension to association rules to accommodate temporal semantics. According to [RR99] logic the technique first search the associationship than it is used to incorporate temporal semantics. It can be used in point based & interval based model of time simultaneously[RR99]. A Frequent pattern approach for mining the time sensitive data was introduced in[CJJX03] Here the pattern frequency history under a tilted-time window framework in order to answer time-sensitive queries. A collection of item patterns along with their frequency histories are compressed and stored using a tree structure similar to FP-tree and updated incrementally with incoming transactions [CJJX03].

3. Problem definition :

3.1 Association Rule:

The concept of association rule, which was motivated by market basket analysis and originally presented by Agrawal. [AS94]. Given a set of T of transaction, an association rule of the form X==> Y is a relationship between the two disjoint itemsets X & Y. An association rule satisfies some user-given requirements. The support of an itemset by the set of transaction is the fraction of transaction that contain the itemset. An itemset is said to be large if its support exceeds a user-given threshold minimum support. The confidence X ==> Y over T is a transaction containing X and also containing Y. Due to complex candidate generation in the data set Jiewai Han invented a new technique of FP-growth method for mining frequent pattern without candidate generation [PH02]. Efficiency of this mining technique is better than all most all algorithm like Apriori, AprioriTid, Apriori Hybrid when data is dense because (1). a large dataset is compressed into a condensed smaller data structure which avoids costly & repeated data scan (2). FP-tree-based mining adopts a pattern-fragment growth method too avoid the costly generation of a large number of candidate generation sets and,(3). A partitioning-based divide- and-conquer method is used to decompose the mining task into a set of similar tasks for

conditional database which dramatically reduce the search space.

In our opinion this mining associationship will be become more useful if we include the time factor in to it.

3.2 Temporal association rule

Definition 1 : The frequency of and itemset over a time period T is the number of transactions in which it occurs divided by total number, of transaction over a time period. In the same way , confidence of a item with another item is the transaction of both items over the period divided by first item of that period.

Support(A) = Frequency of occurrences of A in specified time interval / Total no of Tuples in specified time interval

Confidence(A => B[Ts,Te]) = Support_count(A U B) over Interval / occurrence of A in interval

T_s indicates the valid start time & T_e indicate valid time according to temporal data.

3.3 Simple calendar based Pattern :

When temporal information is applied in terms of date, month , year & week form the term calendar schema. It is introduced in temporal data mining. A calendar schema is a relational schema (in the sense of relational databases) R = (f_n : D_n, F_{n-1} : D_{n-1},……F_1 :d_1) together with a valid constraint. A calendar schema (year : {1995,1996,1997…..} , month : {1,2,3,4,……12}, day : {1,2,3…..31} with the constraint is valid if that evaluates (yy, mm, dd) to True only if the combination gives a valid date. For example <1955,1,3> is a valid date while ,<1996,2,31> is not.

In calendar pattern , the branch e cover e' in the same calendar schema if the time interval e' is the subset of e and they all follow the same pattern. If a calendar pattern <d_n, d_{n-1}, d_{n-2}……..d_1> covers another pattern <d'$_n$, d'$_{n-1;}$, d'$_{n-2}$ ……..d_1> if and only if for each I, 1<=i<=n or d_i = d'$_i$.

Now Our task is to mine frequent pattern over arbitrary time interval in terms of calendar pattern schema.

4 Proposed work-

4.1 Temporal H-Mine (Mem) : Memory -Based Hyper Structure Mining using time dimension.

The problem of Temporal Frequent mining is to find the complete set of item which frequently occurred in valid time interval, for a given support threshold.

This section elaborates how Temporal H-mine process is applied for temporal association rule.

Example 1 Let the first three column of the table be our running transaction id, transaction item and date in which the transaction happened.

Trans Id	Items	Date	Frequent items Projection
100	c,d,e,f,g,i	<*,01,04>	{c,d,e}
200	a,c,d,e,m,b	<*,01,04>	{a,c,d,e,b}
500	a,c,d,e,b	<*,01,04>	{a,c,d,e,b}
400	a,c,d,h	<*,06,04>	{a,d,h}
600	a,b,d,h,i	<*,06,04>	{a,b,d,h}
300	a,b,d,e,g,k	<*,06,04>	{a,b,d,h}

Table 1: Transaction Database

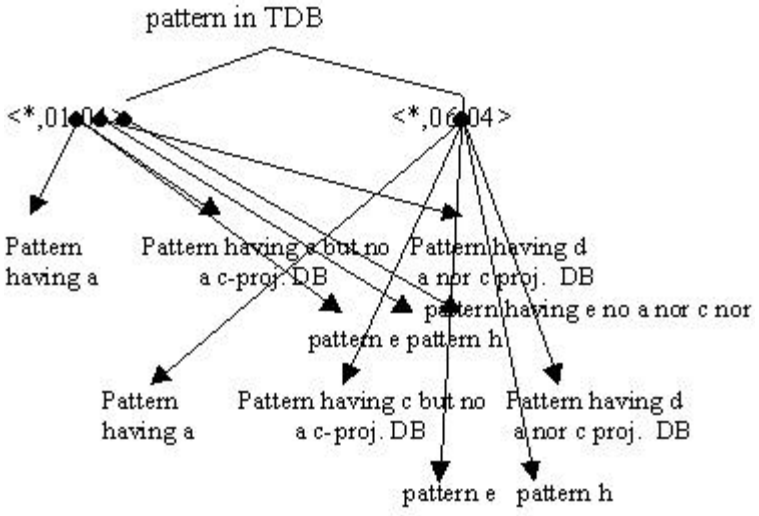


Figure 1: Divide and Conquer method on the basis of time Database

Only the frequent items play roles in frequent pattern mining. A item with lifespan is frequent for some period of time may be infrequent in whole dataset. Due to the pattern change the probability of finding relationship will also be vary in real dataset. By scanning TDB once, the complete set of frequent items on <*,01,04> are {a:2,c:3,d:2,e:3,f:1,g:1,b:2,i:1} and <*,06,01> are{ a:3,b:2,d:3,h:2,e:1,g:1,e:1,k:1 }

A header table H is created separately for each interval, where each frequent item entry has three field : an item-id, a support count and a hyper-link. When the first item projections are loaded into memory, those with the same first item (from DB list) hyper-links as a queue, and entries in header table H act as the head of the

queues. To mine a-projected database ,an a-header table Ha is created. In Ha , every frequent item except for a itself has entry with three field item id, support count and hyper-links & time also [PHNTY01].

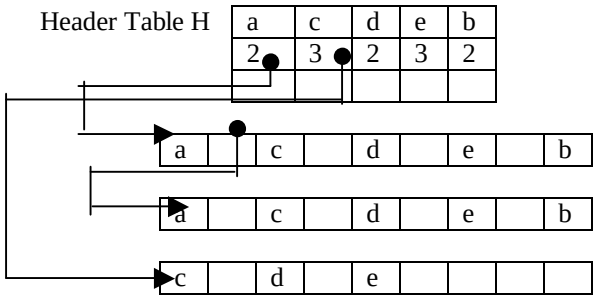


Figure 2 (a). H-struct , the hyper structure for storing frequent -item projection in specific first interval <*,01,04>

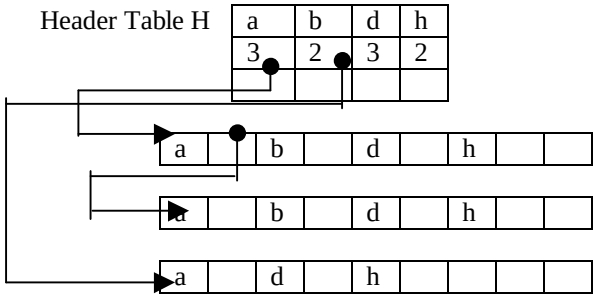


Figure 2 (b): H-struct , the hyper structure for storing frequent -item projection in specific second interval <*,06,04>

Algorithm
Temporal H-mine : The algorithm for memory-based hyper-structure mining.

Input : transaction database TDB with time interval , Support threshold min_sup.

Output : Set of frequent itemset on different time interval.

Method :

1. Scan transaction database TDB once to find L, the complete set of frequent items.
2. Partition TDB into k parts , TDB_1, TDB_2..... TDB_k such that, each TDB_i (i <i <= k) the frequent item projection held in main memory..
3. Check the itemset valid time interval e cover e_0
4. for i = 1 to k, use H-mine (mem) to mine frequent patterns in TDBi with respect to min_sup .
5. Let $F = U^{k}_{i=1} F_i$. Scan TDB one more time, Collect support for frequents in F. Output those patterns which pass the min. supp. on specific time interval.

Although H-mine perform better than Fp-tree in specific cases but Fp-growth method have several advantages over Mining on H-struct since FP-tree shares common prefix path among different transaction., which leads to saving the spaces & time as well. Temporal aspect of data is more important because if a part of data is dense on specific interval may be sparse if we consider the whole dataset. H-mine algorithm is extended when data is dense it call Fp-tree approach for frequent pattern growth method.

4.2 **Handling Dense data sets : Dynamic integration of H-struct and Fp-tree-based mining**

Finding time based frequent pattern in dense database is a challenging task. The FP-growth [PH02] works well known when data is dense. In comparison with FP-growth, Temporal H-mine does not generate physical projected database and conditional Fp-trees and thus save space as well as time in may cases. Temporal Fp-tree based mining has its advantages over mining on H-struct since Temporal Fp-tree shares common prefix paths among different transaction., which may lead to space and time.

The support of dataset in the data warehouse can be maintained by dividing it in different intervals. The support of a item in interval t1 can not be the same in interval t2. A infrequent or less support item in interval t1 can be frequent item in interval t2.

The calendar schema is implemented by applying Apriori algorithm [YPXS03]. It follows the candidate generation approach in order to mine the frequent item. We assist here that instead of candidate generation H-mine and divide & conquer approach is more efficient than Apriori approach. It construct queues for maintains the list of items a tree & each branch indicate the association ship of item. It reduces the size of dataset and increases the performance & efficiency of algorithm. It can solve following queries (1) What are the frequent set over the interval t_1 and t_2 ? (2) what are the period when (a,b) item are frequent ?

(3) Item which are dramatically change from t4 to t1.

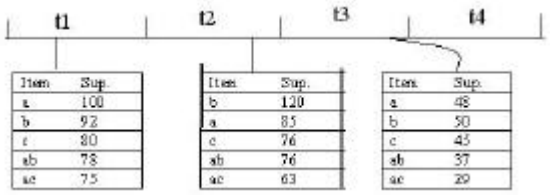


Figure 3 : Frequent pattern in different interval

Lemma 1 :- During transaction in database the association of a item over the support ? can be obtained by projection of the branch of FP-tree.

Rationale : Based on the TFP-tree construction process its frequent item can be projected into a single branch.

For a path a_1, a_2 a_3.....a_k from the root to a node in a FP-tree. Suppose a_{ak} be the count at the node labeled $a_{k\ and}$ c'_{ak} be the sum of the count of the branch of the node.

Tid	Item bought	Date	Calendar pattern	Item in Descending order of Frequency
100	f,a,c,d,g,i,m,p	01/01/2004	<*,01,04>	f,c,a,m
200	a,b,c,f,l,m,o	01/01/2004	<*,01,04>	f,c,a,b,m
300	b,f,h,j,o	02/01/2004	<*,01,04>	f,c,b,o
500	a,f,c,e,l,p,m,n	03/06/2004	<*,06,04>	f,c,a,e,m,l
600	f,a,c,d,e	04/06/2004	<*,06,04>	f,c,a,e
700	b,f,h,j,o,l,o	06/06/2004	<*,06,04>	f,o,l

Table 2. Transaction database in running example

Definition (Temporal FP-Tree) – A Temporal frequent pattern (FP) is tree structure defined below.

It consists of root labeled as "null".

It consist of a set of item-prefix sub trees as the children of the root, and a frequent item header table.

Each node in the item prefix sub tree consists of four fields :

(a) Item name - Item name represents the name of item which is registers on that node (b) count - count registers the number of transactions represented by the portion of the path reaching this node (c) node link - node-link links to the next node of temporal FP-tree (d) calendar pattern time calendar pattern represent the time in which the item transaction appeared.

Each entry in the Frequent –item header table consists of two fields (1) Item name (2) Head of node link

Algorithm : (FP- Tree construction)

Input : A transaction database DB and a minimum support threshold

Output : FP Tree, Fp Tree, Frequent item

Method : The FP tree is constructed ad follows :

1. Scan the database DB once. Collect f, the set of frequent item and support of each item. Sort F from support in descending order as Flist, the list of frequent items
2. Create the root of Temporal FP tree and label it as "Null". For each transaction in DB and do the following

Select the frequent items in Trans and sort them in descending order of Flist. Let the sorted frequent –item list in the Trans be p[P] where p is first element and P is the remaining list. Cal insert_tree(p[P],T).

```
Procedure insert_tree(p[P],T).
{
Step(1) If T has a child N such that
Step(2) if (N.time = P.time) then
// For checking interval phase I
Step(3) if (N.itemname = P. itemname) then
Step(a) N.count = N.count +1
//  Increment the count by 1
Step(4) else create a new node // Node created
//  on the same branch
Step(5) Link to its parent P.count = 1  // Initialize
the counter by 1.
Step(6) else create a new Branch link from the
root.
        Call insert_tree(P,N) recursively

} // End of Function
```

Temporal Frequent pattern Tree : Design & Construction

Let I = {a_1, a_2, a_3.. a_m} be a set of items , and a transaction database DB {T_1, T_2, T_3.....T_n } where T_i {i ? [1..n] } is a transaction which contains a set of items in I.

4.1 TEMPORAL FREQUENT- PATTERN TREE

To design the Temporal FP-tree for frequent pattern mining , let's first examine example from table1.

1. Since time is the most important feature of real world data set, so arrange the item in according to time and define the calendar pattern or interval in calendar unit form.

2. In calendar pattern <*> is used to define that any day or month for example if it used <dd,mm,yy> calendar pattern <*,01,04>

represents any day of Month January & year 2004.

2. Since only the frequent item will play a role in the frequent pattern mining. So first scan is used to identify the set of frequent items
3. If the set of frequent items of each transaction can be stored in some compact data structure , it may be possible to avoid repeatedly scanning the original transaction database.
4. If multiple transaction share a set of frequent items, it may be possible to merge the shared sets with the number of occurrences registered as count. It is easy to check whether two sets are identical if the frequent items in all of the transaction are listed according to a fixed order.
5. If two transaction share a common prefix , according to some sorted order of frequent items, the shared part can be merged into one prefix structure as long as the count is registered properly.

With the above observation , a Temporal frequent pattern tree can be constructed as follows :

First , a scan of DB drives a frequent list of items in schema <*,01,04> are {(f:3),(C:2),(b:2), (a:2),(m:2} same for calendar pattern <*,06,04> frequent items are {(f:3),(c:2),(a:2),(e:2),(m:2)(l:2)} and remaining items are infrequent so skip those item .

Second, the root of the tree is created and labeled with "null". The FP- tree is constructed as follows by scanning the transaction database DB in second time.

1. The scan of the first transaction leads to construction of the first branch of the tree {(f:1),(c:1),(a:1),(m:1),(p:1)}& it follow the calendar pattern <*,01,04>.
2. For the second transaction its temporal period is same so it follow the same branch & the frequent item list {f,c,a,b,m,o} shares a common prefix <f,c,a> the count of each node along the prefix is incremented by 1. and a new node (b:1) is created and linked to child of (a:2), another new one (m:1) is created and linked to as the child of (b:1) and another new one (o:1) is created and linked to as the child of (m:1),
3. For the third transaction its time period is same as previous the transactio is <f,c,b,o> shares common prefix <f,c> so f & c's count is incremented by 1, and a new node b is created although b is already existing but it not go that branch it is not common prefix of the node. Node b is linked as a child of (c:2) and a new node o is created with intialize the count because o is first time introduce on Temporal FP-tree and linked as a child of node <b:1>
4. The scan of forth transaction leader to construct another branch because its time period <*,06,04>does not match with existing branch's node time period. New nodes are created with <(f:1),(c:1),(a:1),(e:1),(m:1),(l:1)>
5. The scan of fifth transaction which follow the time interval of forth transaction so if follow the same branch if the item prefix match, It can share the common prefix <f,c,a,e> with existing path <f,c,a,e,m,l> the count of each node incremented by 1.
6. For the last transaction ,<f,m,l> its time time interval match with second branch so it follow the second branch of FP-tree here it share common prefix f , its count is incremented by 1 , a new node is created for item m , & it is linked to node f by initializing the counter value to1. for next item l again a new node will be created by initializing its counter value

4.3 Mining the frequent item from FP-tree –

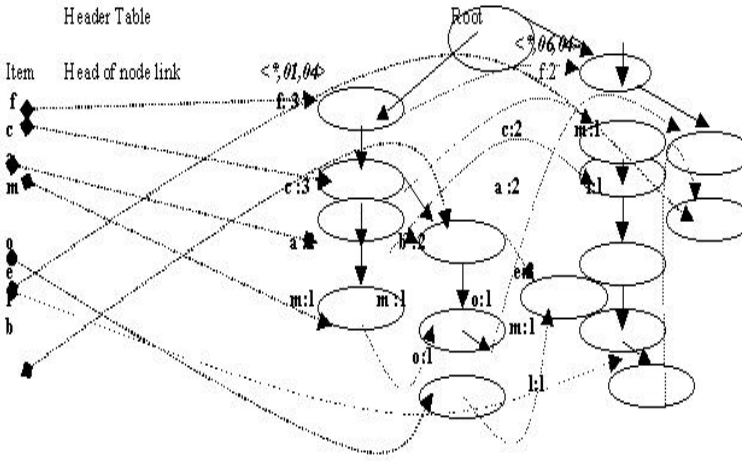


Figure 4 FP tree with different time interval

& it is linked as a child of node m.

In Step(2) if (N.time = P.time) in phase I , its meaning that when a new node p appears in the FP-tree we check the time of transaction , is inside the time of transaction of N item, defined as below table, then it follows the same branch

otherwise a new branch will be created in FP-tree.

Item name	N.Time	P. Time	Branch
F	<01,01,04>		First
C	<01,01,04>	<01,01,04>	First
A	<01,01,04>	<01,01,04>	First
B	<01,06,04>		Second
M	<01,06,04>	<01,06,04>	Second

Table 3. Shows the path depend on the time of transaction in itemset.

Property [PH02] Node Link property . For any frequent item a_i , all the possible patterns contining only frequent items and a_i can be obtained by following a_i, s node link's , starting from a_i 's head in the FP-tree header.

Mining process from the constructed Temporal FP-tree shown in figure1. We examine the mining process by starting from the bottom of node-link header table.
For calendar pattern <*,01,04> for node m ,its intermediate frequent pattern is (m:3), and its path are <f:2,c:2,a:1,m:1,p:1> , <f:1,c:1,b:1,m:1> and <f:1, m:1>. Thus to study which appears together with m at time period <*,01,04> only m prefix {(fca:1) ,(fcb:1),(f:1) }, m's sub -pattern base , which is called m's pattern conditional pattern base.(which is called m's conditional Fp tree) leads to two different branch (fc:3) & (f:2).
For node a , its immediate frequent pattern is (a:4)and it is in two different path one for<*,01,04> & second for <*,06,04>. Calendar pattern <*,01,04> consist of <f:3,c:3 a:2> and <*,06,04> consists of <f:2,c:2>

Item	Time Interval	Conditional Pattern base	Conditional FP-tree
m	<*,01,04>	{(fca:1) ,(fcb:1),(f:1) }	{f:2 c:2|m)
	<*,06,04>	{(fcae:1),(f:1)}	(f:2|m)
a	<*,01,04>	{(fc:2)}	(f:2|a)
	<*,06,04>	{(fc:2)}	(f:2|a)
0	<*,01,04>	{(fcabmo:1),(fcbo:1)}	(f:2,c:2,b:2|o)
	<*,06,04>	ϕ	ϕ
1	<01,*,04>	ϕ	ϕ
	<*,06,04>	{(fcaeml:1),(fml:1)}	(f:2|l)
e	<01,*,04>	ϕ	ϕ
	<*,06,04>	{(fca:2)}	(f:2,c:2 a:2|e)
c	<01,*,04>	{(fc:3)}	(f:2|c)
	<*,06,04>	{(fc:2)}	(f:2|c)
b	<*,01,04>	{(fcb:2)	(f:2,c:2|b)
	<*,06,04>	ϕ	ϕ
f	<01,*,04>	ϕ	ϕ
	<*,06,04>	ϕ	ϕ

Table 3 Mining Temporal frequent patterns by creating conditional (sub) pattern base

From the Temporal FP-tree the conditional frequent pattern tree can be generated by calling the section 3.3 procedure of Frequent pattern–growth method conditionally for every valid interval[PH02].

5. Conclusion & Future work-

In this paper, we have proposed algorithm gives an efficient time sensitive approach for mining frequent item in the dataset. Discovered rule is easier to understand. Temporal H-mine , which takes advantage of H-struct data structure and dynamically adjust link in the mining process.

Temporal Fp-tree, uses divide & conquer technique for construction & traversing of tree which is used to decompose the mining task into a set of smaller task for mining confined pattern in conditional database which dramatically reduce the search space on specific time interval when the data is sparse. H-mine algorithm not need to physically construct memory structures of projected database. In fact Data mining concepts are applied where there are huge set of data available in data warehouse. It requires more scanning & processing time. Hence after applying our logic of the scanning this valid time & processing time can be decreases for mining the frequent set of items. It is very useful for retailer to create its own market strategy as per the requirement of time.

The work can be further extended for designing good classifier and performance can be increases.

Applications:

?? ***Business Application :*** This technique is most useful in Business mining. Most of the real world data have the time varying features. The retailer can change their business policy with time to time for maximize the output Example some model of vehicle are not available from 1980s , suppose is currently appears in the market. Its history indicate no associationship but the fact is that product is not available on that period , so its associationship is started from the interval where it was valid.

?? ***Web Mining :*** The concept can be applicable in web mining , In WWW the site which is no longer available so its associationship also be no longer.

?? ***Clustering Problem :*** This approach can be useful to solve the clustering problem, the cluster can be designed on the basis of period of data., that will reduce the size of data & processing time also.

References –

[AS94] R. Agrawal & R. Srikant , R :" Fast algorithm for mining association rule.In VLDB'94 Chile , Sept 1994,pp –487-499.

[BX00] Claudio Bettini, X. Sean Wang R: " Time Granularies in databases , Data Mining , and Temporal reasoning 2000. pp 230, ISBN 3-540-66997-3, Springer-Verlag, July 2000. 230 pages. Monograph.

[CJJX03] Chris Giannella_, Jiawei Hany, Jian Peiz, Xifeng Yany, Philip S. Yu R: Mining Frequent Patterns in Data Streams at Multiple TimeGranularities, pg 191 – 210, H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds.), Next Generation Data Mining, 2003.

[JG00] Juan M .Ale , Gustavo H. Rossi R: " An approach to discovering temporal association rules", ACM SIGDD March 1..21,2002

[JM01] Jiawei Han, Micheline Kamber , Book : " Data Mining Concept & Technique",2001

[ORS98] Banu Ozden , Sridhar Ramaswamy , Avi Silberschatz R : "Cyclic Association Rule" ,In Proc. Of forteenth International conference on Data Engineering 1998, pp 412-425

[PH02] Jian Pei, Jiawei Han, Yiwen Yin and Running Mao R : Mining Frequent Pattern without Candidate Generation", Kluwer online Academy 2004.

[RHS00] John F. Roddick, Kathleen Hornsby, Myra Spiliopoulou: An Updated Bibliography of Temporal, Spatial, and Spatio-temporal Data Mining Research. TSDM 2000: pp147-164.

[RMS98] S. Ramaswamy, S. Mahajan, and A. Silberschatz. On the discovery of interesting patterns in association rules. In Proc. of the 1998 Int'l Conf. on Very Large Data Bases, pp 368–379, 1998.

[RR99] Chris P. Rainsford, John F. Roddick R: " Adding Temporal semantics to association rule", 3rd International conference KSS Springer 1999, pp 504-509

[YPXS03]Yingjiu Li, Peng Ning, X. Sean Wang , Sushil Jajodia R :"Discovering calendar- based temporal association rules", Data & Knowledge Engineering volume 4,Elesvier publisher, Volume 44 pp– 193-214 ,2003

[PHNTY01] ***J. Pei**, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang, "H-mine : Hyper structure Mining of Freuqent patterns in Large database", In proc. of International conference on Data Mining , San Jose, California, Novermber 29-December 2, 2001.*

Artemis Message Exchange Framework: Semantic Interoperability of Exchanged Messages in the Healthcare Domain *

Veli Bicer, Gokce B. Laleci, Asuman Dogac, Yildiray Kabak
Software Research and Development Center
Middle East Technical University (METU)
06531 Ankara Türkiye
email: asuman@srdc.metu.edu.tr

ABSTRACT

One of the most challenging problems in the healthcare domain is providing interoperability among healthcare information systems. In order to address this problem, we propose the semantic mediation of exchanged messages. Given that most of the messages exchanged in the healthcare domain are in EDI (Electronic Data Interchange) or XML format, we describe how to transform these messages into OWL (Web Ontology Language) ontology instances. The OWL message instances are then mediated through an ontology mapping tool that we developed, namely, OWLmt. OWLmt uses OWL-QL engine which enables the mapping tool to reason over the source ontology instances while generating the target ontology instances according to the mapping patterns defined through a GUI.

Through a prototype implementation, we demonstrate how to mediate between HL7 Version 2 and HL7 Version 3 messages. However, the framework proposed is generic enough to mediate between any incompatible healthcare standards that are currently in use.

1. INTRODUCTION

Most of the health information systems today are proprietary and often only serve one specific department within a healthcare institute. A number of standardization efforts are progressing to address this interoperability problem such as EHRcom [3], openEHR [15] and HL7 Version 3 [6]. Yet, it is not realistic to expect all the healthcare institutes to conform to a single standard. Furthermore, different versions of the same standard (such as HL7 Version 2 and Version 3) and even the different implementations of the same standard, for example, some HL7 Version 2 implementations, do not interoperate. Therefore there is a need to address the interoperability problem at the semantic level. Semantic interoperability is the ability for information shared by systems to be understood at the level of formally defined domain concepts so that the information is computer processable by the receiving system [10].

In this paper, we describe an engineering approach developed within the scope of the Artemis project [1] to provide the exchange of meaningful clinical information among healthcare institutes through semantic mediation. The proposed framework, called AMEF (Artemis Message Exchange Framework) involves first providing the mapping of a source ontology into a target ontology with the help of a mapping tool which produces a mapping definition. This mapping definition is then used to automatically transform the source ontology message instances into target message instances.

Through a prototype implementation, we demonstrate how to mediate between HL7 Version 2 and HL7 Version 3 messages. However, the framework proposed is generic enough to mediate between any incompatible healthcare standards that are currently in use.

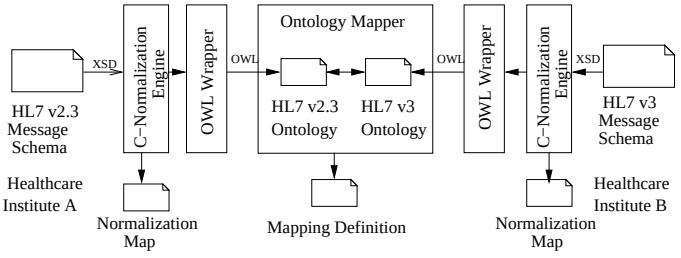


Figure 1: Message Schema Mapping Process

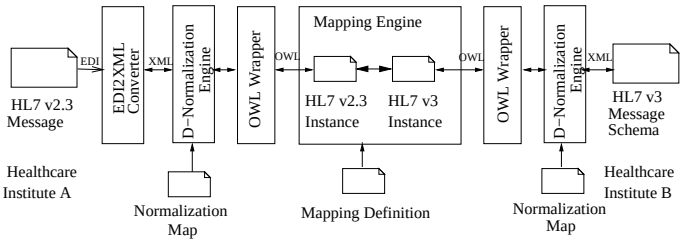


Figure 2: Automatic Message Instance Transformation Process

The semantic mediation between HL7 Version 2 and HL7 Version 3 messages is realized in two phases:

- *Message Ontology Mapping Process*: In the first phase, the message ontologies of two healthcare institutes are mapped one another (Figure 1). Assume that healthcare institute A uses HL7 v2 and healthcare institute B uses HL7 v3 to provide system interconnection. The message ontologies of these institutes are mapped one into other by using an ontology mapping tool. For this

*This work is supported by the European Commission through IST-1-002103-STP Artemis project and in part by the Scientific and Technical Research Council of Turkey (TÜBİTAK), Project No: EEEAG 104E013

purpose we have developed an OWL (Web Ontology Language) ontology mapping tool, namely, OWLmt [16]. With the help of a GUI, OWLmt allows to define semantic mappings between structurally different but semantically overlapping OWL ontologies, and produces a "Mapping Definition".

Since message ontologies for HL7 messages do not exist yet, we use the HL7 Version 2 and Version 3 XML Schemas (XSDs) [19] to generate OWL ontologies. This process, called "Conceptual Normalization" [5] produces a "Normalization map" describing how a specific message XSD is transformed into the corresponding OWL schema.

The "Mapping Definitions" and the "Normalization map" produced in the first phase are used during the second phase to automatically transform the message instances one into another.

- *Message Instance Mapping*: In the second phase (Figure 2), first the XML message instances of healthcare institute A are transformed into OWL instances by using the "Data Normalization" engine [5]. Note that if the message is in EDI (Electronic Data Interchange) format, it is first converted to XML. Then by using the *Mapping definition*s, OWL source (healthcare institute A) messages instances are transformed into the OWL target (healthcare institute B) message instances. Finally the OWL messages are converted to XML again through the "Data Normalization" engine.

RQC Request Clinical Information	
MSH	Message Header
QRD	Query Definition
[QRF]	Query Filter
{	
PRD	Provider Data
[{ CTD }]	Contact Data
}	
PID	Patient Identification
[{ NK1 }]	Next of Kin/Associated Parties
[{ GT1 }]	Guarantor
[{ NTE }]	Notes and Comments

RCI Return Clinical Information	
MSH	Message Header
MSA	Message Acknowledgment
[QRF]	Query Filter
{	
PRD	Provider Data
[{ CTD }]	Contact Data
}	
PID	Patient Identification
[{ DG1 }]	Diagnosis
[{ DRG }]	Diagnosis Related Group
[{ AL1 }]	Allergy Information
[	
{	
OBR	Observation Request
[{ NTE }]	Notes and Comments
[	
{	
OBX	Observation Result
[{ NTE }]	Notes and Comments
}	
]	
}	
]	
[{ NTE }]	Notes and Comments

Figure 3: The Structures of the RQC/RCI EDI messages for the HL7 Version 2 event I05

The paper is organized as follows: In Section 2, we briefly summarize the HL7 standard. Section 3 describes the semantic mediation of HL7 v2 and v3 messages. The details of OWL mapping tool used in the mediation is presented in Section 4. Transforming HL7 v2 EDI messages to XML is briefly introduced in Section 5. Finally Section 6 describes the "Normalization" tool used and the improvements realised on this tool.

2. HEALTH LEVEL 7 (HL7) STANDARD

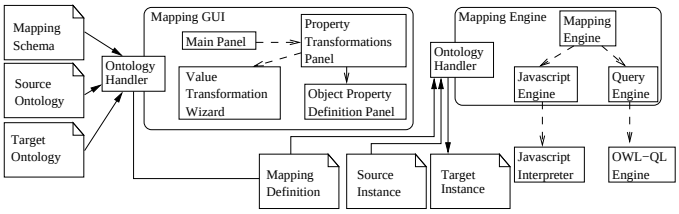


Figure 4: Architecture of OWLmt

The primary goal of HL7 is to provide standards for the exchange of data among healthcare computer applications. The standard is developed with the assumption that an event in the healthcare world, called the *trigger event*, causes exchange of messages between a pair of applications. When an event occurs in an HL7 compliant system, an HL7 message is prepared by collecting the necessary data from the underlying systems and it is passed to the requestor, usually as an EDI message. For example, as a result of a trigger event, say "I05", the clinical patient information for a given patient identifier is passed to the requestor as shown in Figure 3. Clinical information refers to the data contained in a patient record such as problem lists, lab results, current medications, family history, etc. [7].

HL7 version 2 is the most widely implemented healthcare informatics standard in the world today. Yet being HL7 Version 2 compliant does not imply direct interoperability between healthcare systems. Version 2 messages, contain many optional data fields. For example every attribute presented in square brackets in Figure 3, denotes optional information that may be omitted. This optionality provides great flexibility, but necessitates detailed bilateral agreements among the healthcare systems to achieve interoperability.

To remedy this problem, HL7 has developed Version 3 [6] which is based on an object-oriented data model, called Reference Information Model (RIM) [8]. The main objective of the HL7 Version 3 is to eliminate the optionality. RIM is used as the source of the content of messages and this results in a more efficient message development process. The result of the Version 3 process is the Hierarchical Message Definition (HMD), which defines the schema of the messages based on the RIM classes. Note that HL7 Version 3 messages do not interoperate with HL7 Version 2 messages.

3. SEMANTIC MEDIATION OF HL7 V2 AND V3 MESSAGES

In Artemis Message Exchange Framework (AMEF), the semantic mediation of HL7 v2 and v3 messages is realized in two phases:

- *Message Schema Mapping Process*: In the first phase, the message schemas of two healthcare institutes are mapped one another through semantic mediation as shown in Figure 1. At the heart of this process is the OWL Mapping tool, OWLmt, transforming OWL ontologies one into other.

  The OWL ontologies corresponding to the message schemas involved are generated through a set of available tools. First, for healthcare institute A (Figure 1), the HL7 Version 2 XML Schemas (XSDs) [19] are

converted to RDFS (Resource Description Framework Schema) by using the Conceptual Normalization (C-Normalization) engine of the Harmonise project [5]. This process uses a set of heuristics as described in Section 6 and produces a "Normalization map" describing how a specific HL7 Version 2 message XSD is transformed into the corresponding RDFS schema and vice-versa. Then, by using the OWL Wrapper, which we developed using Jena API [11], RDFS Schemas are transformed to OWL.

On the other hand, for healthcare institute B (Figure 1), in order to generate the XSDs of HL7 v3 messages, RoseTree tool of HL7 is used [18]. RoseTree allows the user to graphically build a HMD (Hierarchical Message Definition) from the Reference Information Model of HL7 v3. This generated HMD file describes the structure of the v3 XML messages, but it is not in XSD format. In order to translate the HMD file to XSD, "HL7 v3 Schema Generator" [9] is used.

The next step is to map the source ontology into the target ontology by using OWLmt. This process is described in detail in Section 4.

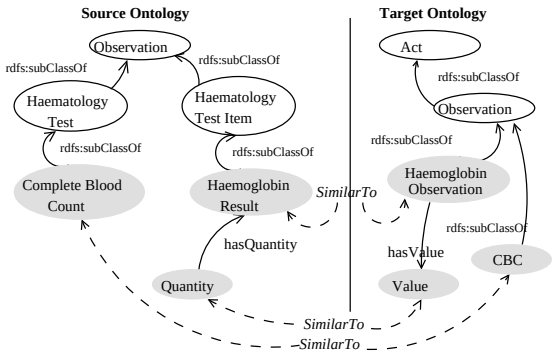


Figure 5: Mapping between HL7 v2 and HL7 v3 message structures

- *Message Instance Mapping*: In the second phase (Figure 2), first the HL7 version 2 EDI messages are converted to XML. The open-source programming library from HL7, namely, HL7 application programming interface (HAPI) [4] is used for transforming the EDI messages into their XML representations.

  In the next step, as shown in Figure 2, the XML message instances of healthcare institute A are transformed to OWL instances by the "Data Normalization (D-Normalization) engine [5] using the "Normalization map" produced during the first phase.

  Then by using the *Mapping definition*s, OWLmt transforms OWL source (healthcare institute A) messages instances into the OWL target (healthcare institute B) message instances. Finally the OWL messages are converted to the XML format that the healthcare institute B understands, again through the "Data Normalization" engine as shown in Figure 2.

In the following sections, we describe how these tools realize the described functionality.

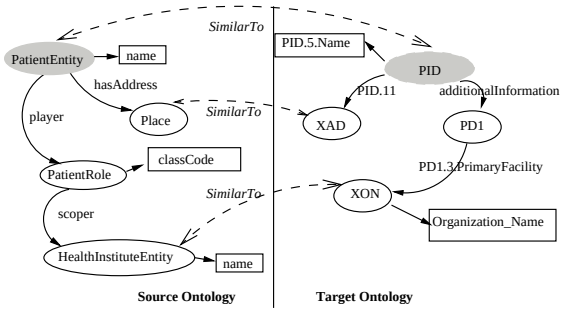


Figure 6: Mapping Object properties

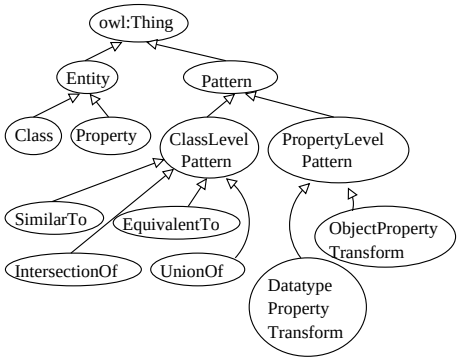


Figure 7: OWL Mapping Schema

4. OWL MAPPING TOOL: OWLMT

We have developed an OWL mapping tool, called OWLmt, to handle ontology mediation by mapping the OWL ontologies in different structures and with an overlapping content one into other. The architecture of the system, as shown in Figure 4, allows mapping patterns to be specified through a GUI tool based on a Mapping Schema. The Mapping Schema, as shown in Figure 7, is also defined in OWL.

Mapping patterns basically involve the following:

- *Matching the source ontology classes to target ontology classes*: In order to represent the matching between the classes of source and target ontologies, we have defined four mapping patterns: *EquivalentTo*, *SimilarTo*, *IntersectionOf* and *UnionOf*. Two identical classes are mapped through *EquivalentTo* pattern. *SimilarTo* implies that the involved classes have overlapping content. How the similar classes are further related is detailed through their data type properties and object properties by using "property mapping patterns". As an example, in Figure 5, the "HaemeglobinResult" class in HL7 v2 ontology is defined to be similar to "HaemoglobinObservation" class in HL7 v3 ontology. The mappings of the "hasQuantity" and "hasValue" object properties of these classes are handled by defining an "ObjectPropertyTransform" pattern between these properties.

  The *IntersectionOf* pattern creates the corresponding instances of the target class as the intersection of the declared class instances. Similarly, the UnionOf pat-

tern implies the union of the source classes' instances to create the corresponding instances of the target class. Furthermore, a class in a source ontology can be a more general (super class) of a class in the target ontology. In this case, which instances of the source ontology makes up the instances of the target ontology is defined through KIF (Knowledge Interchange Format) [13] conditions to be executed by the OWLmt mapping engine. When a source ontology class is a more specific (sub class) of a target ontology class, all the instances of the source ontology qualify as the instances of the target ontology.

- *Matching the source ontology Object Properties to target ontology Object Properties*: In addition to matching a single object property in the source ontology with a single object property in the target ontology, in some cases, more than one object properties in the source ontology can be matched with one or more object properties in the target ontology. Consider the example given in Figure 6. According to the HL7 v3 specifications, two entities, "Patient" and "HealthInstitueEntity" are connected by a "Role" class which is "PatientRole" in this case. On the other hand, in the target ontology, the "XON" class in HL7 v2.x represents the healthcare facility that a patient is registered. "PD1" (Patient Demographics 1) gives the patient information. "XON" is connected to the "PD1" by the "PD1.3.PrimaryFacility" object property. As it is clear from this example, relating a single object property in source ontology with a single object property in the target ontology does not suffice: There may be paths consisting of object property relations in the source and target ontologies that need to be mapped.

  OWLmt allows defining "ObjectPropertyTransform" pattern which represents the path of classes connected through object properties such that whenever a path defined in the source ontology (inputPath) is encountered in the source ontology instance, the path defined for target ontology (outputPath) is created in the target ontology instance. Paths are defined as triples in KIF [13] format and executed through the OWL-QL [17] engine. For example, assuming the path defined in the source ontology (Figure 6):

  (rdf:type ?x PatientEntity) (player ?x ?y)

  (rdf:type ?y PatientRole) (scoper ?y ?z)

  (rdf:type ?z HealthInstituteEntity).

  and assuming that it corresponds to the following path in the target ontology:

  (rdf:type ?x PID) (additionalInformation ?x ?y) (rdf:type ?y PD1) (PD1.3.PrimaryFacility ?y ?z) (rdf:type ?z XON)

  OWLmt constructs the specified paths among the instances of the target ontology in the execution step based on the paths defined among the instances of the source ontology.

- *Matching source ontology Data Properties to target ontology Data Properties*: Specifying the "DatatypePropertyTransform" helps to transform data type properties of an instance in the source ontology to the corresponding data type properties of instance in the target ontology. Since the data type properties may be structurally different in source and target ontologies, more complex transformation operations may be necessary than copying the data in source instance to the target instance. XPath specification [20] defines a set of basic operators and functions which are used by the OWLmt such as "concat", "split", "substring", "abs", and "floor". In some cases, there is a further need for a programmatic approach to specify complex functions. For example, the use of conditional branches (e.g. if-then-else, switch-case) or iterations (e.g while, for-next) may be necessary in specifying the transformation functions. Therefore, we have added JavaScript support to OWLmt. By specifying the JavaScript to be used in the "DatatypePropertyTransform" pattern, the complex functions can also be applied to the data as well as the basic functions and the operators provided by XPath.

4.1 OWLmt Mapping Schema

The mapping patterns used in the OWLmt are defined through an OWL ontology called "Mapping Schema". Each mapping pattern is an owl:class in the "Mapping Schema" as shown in Figure 7. The additional information needed in the execution of the patterns are provided as KIF [13] expressions such as *inputPath*s and *outputPath*s. The *inputPath* and *outputPath* are data type properties of "ObjectProperty Transform Pattern" class and hold the query strings in the KIF format which are used in the execution to query the source ontology instances in order to build the target instances.

Each mapping relation specified through OWLmt GUI represents as an instance of these pattern classes, and the final the mapping definition is stored as an instance of the "Mapping Scheme" as a collection of pattern class instances.

In Figure 8, a part of the mapping definition of the example in Figure 6 is presented. First the *SimilarTo* relationship between the "Patient" and "PatientEntity" classes are represented with an instance of *SimilarTo* pattern. Then through an "ObjectPropertyTransform" pattern instance, the relationships between object properties linking the "PatientEntity" to "HealtInstituteEntity" classes and the object property linking the "PID" to "XON" classes are represented. Further details of the mapping tool are presented in [2].

This mapping definition is given as an input to the OWLmt Mapping Engine, which translates source ontology instances to target ontology instances.

4.2 OWLmt GUI

OWLmt GUI [16] consists of five components: Ontology Handler, Main Panel, Property Transformations Panel, Value Transformation Wizard and Object Property Definition Panel. The Ontology Handler is used in parsing and serializing the ontology documents. The class mapping patterns are defined in the main panel. The property mapping patterns are defined in the property transformation panel. This panel lets the user to create new property mapping patterns such as the "ObjectPropertyTransform" and "DatatypePropertyTransform". The value transformation wizard is used to configure a "DatatypePropertyTransform" pattern. By using this wizard, the functions used in the value transformation of the data type properties can be spec-

```
<SimilarTo rdf:ID="SimilarTo_1">
 <similarToInput>
    <relatedTo rdf:resource=#PatientEntity/>
 </similarToInput>
 <similarToOutput>
    <relatedTo rdf:resource=#PID/>
 </similarToOutput>
 <operationName>PatientEntity_SimilarTo_PID</operationName>
</SimilarTo> .......

<ObjectPropertyTransform rdf:ID="ObjectPropertyTransform_1">
  <operationName>ObjectPropertyTransform_1</operationName>
  <includedIn rdf:resource=#SimilarTo_1/>
  <inputPath>(rdf:type ?x PatientEntity) (player ?x ?y)
       (rdf:type ?y PatientRole)   (scoper ?y ?z)
        (rdf:type ?z HealthInstituteEntity)
  </ inputPath>
  <outputPath>(rdf:type ?x PID) (additionalInformation ?x ?y)
          (rdf:type ?y PD1) (PD1.3.PrimaryFacility ?y ?z)
          (rdf:type ?z XON)
  </ outputPath>
</ObjectPropertyTransform>
```

Figure 8: An Example Mapping Definition

ified.

4.3 OWLmt Engine

The mapping engine is responsible for creating the target ontology instances using the mapping patterns given in the Mapping Definition and the instances of the source ontology. It uses OWL Query Language (OWL-QL) to retrieve required data from the source ontology instances. OWL-QL is a query language for OWL developed at the Stanford University [17]. While executing the class and property mapping patterns, the query strings defined through the mapping GUI are send to the OWL-QL engine with the URL of the source ontology instances. The query engine executes the query strings and returns the query results.

The OWL-QL engine uses the JTP (Java Theorem Prover) reasoning engine [12], an object-oriented modular reasoning system. The modularity of the system enables it to be extended by adding new reasoners or customizing existing ones.

The use of the OWL-QL enables OWLmt to have reasoning capabilities. When querying the source ontology instances or while executing the KIF [13] patterns, OWL-QL reasons over the explicitly stated facts to infer new information. As an example, consider two instances, I1 and I2, which are the members of the classes C1 and C2 respectively. If these two instances are related with the "owl:sameAs" construct, one of them should be in the extension of the intersection class, say C3, of the classes C1 and C2. Hence, the *IntersectionOf* pattern transforms the instance I1 and I2 to the instance I3 which is a member of C3 in the target ontology. However, assume that there is no direct "owl:sameAs" construct but there is a functional property which implies that these two instances are the same. The reasoning engine can infer from the definition of the "owl:FunctionalProperty" by using the rule:

```
(rdf:type ?prop owl:FunctionalProperty)
(?prop ?instance ?I1)
(?prop ?instance ?I2)
->
(owl:sameAs ?I1 ?I2)
```

that the instances I1 and I2 are the same instance resulting in the instance I3 to be in the target ontology.

After executing the class mapping patterns, the mapping engine executes the property mapping patterns. Similar to the class mapping patterns, OWL-QL queries are used to locate the data. In order to perform value transformations, the mapping engine uses the JavaScripts in the "DatatypePropertyTransform" pattern. To execute the JavaScripts, an interpreter is used. The engine prepares the JavaScript by providing the values for the input parameters and sends it to the interpreter. The interpreter returns the result, which is then inserted as the value of the data type property in the target ontology instance.

5. EDI TO XML CONVERSION IN HL7

There are several commercial and open-source programming libraries that implement the HL7 standards. In our implementation, HAPI [4] (HL7 Application Programming Interface) Assembler/Disassembler Tool is used to transform the HL7 v2 EDI messages into their XML representations. HAPI provides open source libraries for parsing and manipulating both EDI and XML messages that are HL7 conformant. Furthermore the library enables message validation, that is, enforcement of HL7 data type rules for the values in the messages.

6. NORMALIZATION TOOL

As previously mentioned, currently the healthcare application messages are usually in XML or EDI format (which can be converted to XML). Hence there is a need for automatic bidirectional transformation of XML message instances to OWL message instances as well as automatic generation of OWL Schemas from XML Schema Definitions (XSDs). Such a transformation, called Normalization, has been realized within the scope of the Harmonise project [5].

The first step in the "Normalization" process is generating RDFS schemas from local XSD schemas. This step is called Conceptual Normalization (C-Normalization) phase where the C-Normalization engine parses the XML Schema, and using a set of predefined "Normalization Heuristics", creates the corresponding RDFS schema components for each XML Schema component automatically. Normalization Heuristics define how specific XML Schema construct can be projected onto a RDFS construct (entity or set of related entities) [5]. With this process, the complex type, element and attribute definitions of the XSD are represented as classes, and properties in the RDFS ontology. One of the "Normalization Heuristics" called "ComplexType2Class" projects each complex type definition in XSD onto a class definition in RDFS. Furthermore, the attribute definitions and the element definitions in XSD are converted to the "rdf:Property" by the "Attribute2Property" and "Element2Property" heuristics, respectively. After representing the complex types as classes and elements as properties, the domain and range of the properties are set. The "ElementParent2PropertyDomain" heuristic sets the domain of the property to the class which corresponds to the parent of the element in the XSD. Furthermore, the "ElementType2PropertyRange" heuristic sets the range of the property to the class which corresponds to the type of the element in the XSD as illustrated in Figure 9. The C-Normalization process produces a "Normalization Map" which defines the associations between the XML Schema and the re-engineered RDFS model. Further details

of this work are available in [5].

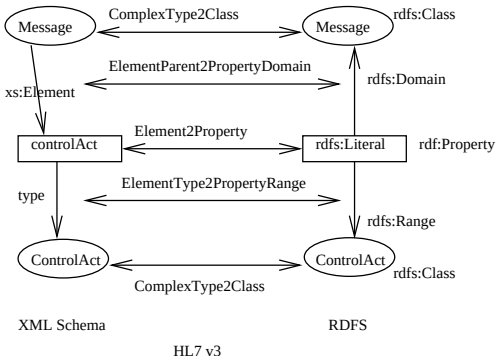


Figure 9: C-Normalization Phase

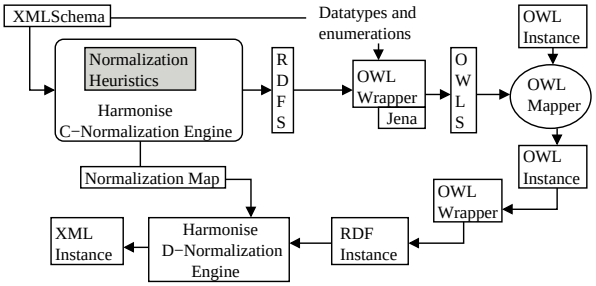


Figure 10: Normalization process for the bidirectional transformation of XML instances to OWL instances

The second step in "Normalization" is the Data Normalization Process (D-Normalization) which is used for transforming the data instances from XML to OWL or OWL to XML.

In Artemis architecture, we have used the Harmonise Normalization Engine. However since we need OWL Schemas instead of RDFS schemas, we developed an OWL wrapper using Jena API to create OWL schemas from the RDFS files after the C-Normalization step. Additionally in the D-Normalization step, through the same wrapper, the generated RDF instances are further translated in to OWL instances or vice versa as depicted in Figure 10.

Note that in Harmonise C-Normalization step, the enumeration of property values or basic data types defined in XML Schemas cannot be preserved. To handle this, the OWL Wrapper developed carries the enumeration of property values and basic data types to the OWL Schema. The enumerated classes are represented using <owl:oneOf rdf:parseType="Collection"> construct in case of enumerated classes, and using <owl:oneOf> and <rdf:List> constructs in case of enumerated data types. The data types are represented by referring to XML Schema data types using RDF data typing scheme.

7. CONCLUSIONS

One of the most challenging problems in the healthcare domain today is providing interoperability among healthcare information systems. In order to tackle this problem, we propose an engineering approach to semantic interoperability within the scope of the Artemis project. For this purpose, the existing applications are wrapped as Web services and the messages they exchange are annotated with OWL ontologies which are then mediated through an ontology mapping tool developed, namely, OWLmt. One of the major contributions of the OWLmt is the use of OWL-QL engine which enables the mapping tool to reason over the source ontology instances while generating the target ontology instances according to the graphically defined mapping patterns.

8. REFERENCES

[1] Artemis A Semantic Web Servicebased P2P Infrastructure for the Interoperability of Medical Information Systems, http://www.srdc.metu.edu.tr/-webpage/projects/artemis/.

[2] Bicer, V., "OWLmt: OWL Mapping Tool", M.Sc. Thesis, Dept. of Computer Eng., METU, in preparation.

[3] ENV 13606:2000 "Electronic Healthcare Record Communication", http://www.centc251.org/TCMeet/-doclist/TCdoc00/N00048.pdf.

[4] HL7 Application Programming Interface (HAPI), http://hl7api.sourceforge.net

[5] Harmonise, IST200029329, Tourism Harmonisation Network, Deliverable 3.2 Semantic mapping and Reconciliation Engine subsystems.

[6] Health Level 7, http://www.hl7.org.

[7] HL7, Chapter 11 Patient Referral, http://www.hl7.org/library/General/v231.zip

[8] HL7 Reference Information Model (RIM), http://www.hl7.org/library/data-model/RIM/-modelpage_mem.htm.

[9] HL7 v3 Schema Generator, http://www.hl7.org/-library/data-model/V3Tooling/toolsIndex.htm

[10] ISO/TS Health Informatics - Requirements for an electronic health record architecture, Technical Specification, International Organization for Standardization (ISO), Geneva, Switzerland, 2004.

[11] Jena Framework, http://jena.sourceforge.net/ .

[12] Java Theorem Prover (JTP), http://www.ksl.stanford.edu/software/JTP/ .

[13] Knowledge Interchange Format (KIF), http://logic.stanford.edu/kif/kif.html .

[14] A. Maedche, D. Motik, N. Silva, R. Volz, "MAFRA-A MApping FRAmework for Distributed Ontologies", In Proc. of the 13th European Conf. on Knowledge Engineering and Knowledge Management EKAW-2002, Madrid, Spain, 2002.

[15] OpenEHR Foundation, http://www.openehr.org/ .

[16] OWL Mapping Tool (OWLmt), http://www.srdc.metu.edu.tr/artemis/owlmt/

[17] OWL Query Language, http://ksl.stanford.edu/projects/owlql/

[18] Rose Tree, http://www.hl7.org/library/data-model/-V3Tooling/toolsIndex.htm

[19] XML encoding rules of HL7 v2 messages - v2.xml, http://www.hl7.org/Special/Committees/xml/drafts/-v2xml.html

[20] XML Path Language, http://www.w3.org/TR/xpath .

Database Research at Bilkent University

Özgür Ulusoy
Bilkent University
Computer Engineering Department
Bilkent, Ankara, TURKEY
oulusoy@cs.bilkent.edu.tr

1. Introduction

This report provides a brief description of the research activities of the Database Research Group of Bilkent University. The current research of the group is mainly focused on the topics of *Multimedia Databases* (in particular video database management and content-based retrieval of document images), *Web Databases* (in particular the modeling and querying of Web resources and focused crawling), and *Mobile Computing* (in particular moving object processing and mobile data management).

An overview of each research topic investigated is provided together with the list of researchers, any external funding information, and a selection of associated publications. Interested readers should contact Özgür Ulusoy at oulusoy@cs.bilkent.edu.tr for further information. Most of the publications listed in this report are available at:
http://www.cs.bilkent.edu.tr/~oulusoy/~pubs.html.

2. Multimedia Databases

Researchers: Ö. Ulusoy, U. Güdükbay, E. Çetin, E. Şaykol, C. Alper, I. S. Altıngövde, T. Sevilmiş.
Past Researchers: M. E. Dönderler, U. Arslan, G. Ünel, A. K. Sinop.

Funding Sources: Scientific and Technical Research Council of Turkey (TÜBİTAK) under grant number EEEAG-199E025, Turkish State Planning Organization (DPT) under grant number 2004K120720, and European Commission 6th Framework Program, MUSCLE NoE Project, under grant number FP6-507752.

2.1 BilVideo: A Video Database Management System

We have developed a prototype video database management system, called *BilVideo* [1, 2, 3]. The architecture of *BilVideo* is original in that it provides full support for spatio-temporal queries that contain any combination of directional, topological, 3D-relation, object-appearance, trajectory-projection, and similarity-based object-trajectory conditions by a rule-based system built on a knowledge-base, while utilizing an object-relational database to respond to semantic (keyword, event/activity, and category-based), color, shape, and texture queries. The knowledge-base of *BilVideo* consists of a fact-base and a comprehensive set of rules implemented in Prolog. The rules in the knowledge-base significantly reduce the number of facts that need to be stored for spatio-temporal querying of video data [4].

To respond to user queries containing both spatio-temporal and semantic conditions, the query processor interacts with both the knowledge-base and a feature database, where the system stores fact-based and semantic metadata, respectively. Intermediate query results returned from these two system components are integrated seamlessly by the query processor, and final results are sent to Web clients. Raw video data and its features are stored in a separate database. The feature database contains video semantic properties to support keyword, event/activity, and category-based queries. The *video-annotator tool*, which we developed as a Java application, generates and maintains the features. The fact-base, which is a part of the knowledge-base, is populated by the *fact-extractor tool*, which is also a Java application [3].

BilVideo provides support for retrieving any segment of a video clip, where the given query conditions are satisfied, regardless of how video data is semantically partitioned. Object trajectories, object-appearance relations, and spatio-temporal relations between video objects are represented as Prolog facts in a knowledge-base, and they are not explicitly related to the semantic units of videos. Thus, precise answers can be returned for user queries, when requested, in terms of frame intervals.

BilVideo has a simple, yet very powerful SQL-like query language, which currently supports a broad range of spatio-temporal queries on video data [5]. We are currently working on integrating support for semantic and low-level (color, shape, and texture) video queries as well. We completed our work on semantic video modeling, which was reported in [6]. As for the low-level queries, our *Fact-Extractor* tool also extracts color, shape, and texture histograms of the salient objects in video keyframes. We have also developed a Web-based visual query interface for specifying video queries visually over the Internet. Furthermore, we have completed our work on the optimization of spatio-temporal video queries [7].

The *BilVideo* query language is designed to be used for any application that needs video query processing facilities. Hence, the language provides query support through external predicates for application-dependent data.

The Web-based Query Interface of BilVideo and its user manual are available at http://pcvideo.cs.bilkent.edu.tr/. A demo of the Web-based Query Interface can be seen at:
http://www.cs.bilkent.edu.tr/~bilmdg/bilvideo/webclient.avi.

2.2 Ottoman Archive Content-Based Retrieval System

The Ottoman Archive Content-Based Retrieval system is a Web-based program that provides electronic access to digitally stored

Ottoman document images. The Ottoman script is a connected script based on the Arabic alphabet. A typical word consists of compounded letters as in handwritten text. We have developed a framework for content-based retrieval of historical documents in the Ottoman Empire archives [8]. The documents are stored as textual images, which are compressed by constructing a library of symbols that occur in a document. The symbols in the original image are then replaced by the pointers into the codebook library. For symbol extraction, we use the features in wavelet and spatial domain based on angular and distance span shapes. Symbols are extracted from document images by a scale-invariant process. User can specify a query as a rectangular region in an input image, and content-based retrieval is achieved by applying the same symbol extraction process to the query region. The query is processed on the codebook of documents, and the resulting documents are ranked in the decreasing order of their similarity to the query image, which is determined by the total number of symbols matched with the query region. The resulting documents are presented by identifying the matched region of each document in a rectangle. The querying process does not require decompression of images.

The techniques we use in our work are not specific to documents with Ottoman script. They can easily be tailored to other domains of archives containing printed or handwritten documents.

The web site of the Ottoman Archive Content-Based Retrieval system is:
http://www.cs.bilkent.edu.tr/bilmdg/ottoman/webclient.html.

A step-by-step user manual describing how to access the Web query interface and specify queries is available at:
http://www.cs.bilkent.edu.tr/~bilmdg/ottoman/manual/manual.htm.

References

[1] M.E. Dönderler, E. Saykol, Ö. Ulusoy, U. Güdükbay. BilVideo: A Video Database Management System. *IEEE Multimedia*, 10, 5 (January/March 2003), 66-70.

[2] Ö. Ulusoy, U. Güdükbay, M.E. Dönderler, E. Saykol, C. Alper. BilVideo Video Database Management System (demo paper). In *Proceedings of the International Conference on Very Large Databases (VLDB'04)*. (Toronto, Canada, August-September 2004), 1373-1376.

[3] M.E. Dönderler, E. Saykol, U. Arslan, Ö. Ulusoy, U. Güdükbay. BilVideo: Design and Implementation of a Video Database Management System. To appear in *Multimedia Tools and Applications*, 2005.

[4] M.E. Dönderler, Ö. Ulusoy, U. Güdükbay. A Rule-based Video Database System Architecture. *Information Sciences*, 143, 1-4 (June 2002), 13-45.

[5] M.E. Dönderler, Ö. Ulusoy, U. Güdükbay. Rule-based Spatio-temporal Query Processing for Video Databases. *VLDB Journal*, 13, 1 (January 2004), 86-103.

[6] U. Arslan, M.E. Dönderler, E. Şaykol, Ö Ulusoy, U. Güdükbay. A Semi-Automatic Semantic Annotation Tool for Video Databases. In *Proceedings of the Workshop on Multimedia Semantics (SOFSEM'02)*. (Milovy, Czech Republic, November 2002), 1-10.

[7] G, Ünel, M.E. Dönderler, Ö. Ulusoy, U. Güdükbay. An Efficient Query Optimization Strategy for Spatio-Temporal Queries in Video Databases. *Journal of Systems and Software*, 73, 1 (September 2004), 113-131.

[8] E. Şaykol, A. K. Sinop, U. Güdükbay, Ö. Ulusoy, E. Çetin. Content-Based Retrieval of Historical Ottoman Documents Stored as Textual Images. *IEEE Transactions on Image Processing*, 13, 3 (March 2004), 314-325.

3. Web Databases

Researchers: Ö. Ulusoy, İ. S. Altıngövde, Ö. N. Subakan.
Past Researchers: S. A. Özel, M. Kutlutürk.
Collaborators: G. Özsoyoğlu, Z. M. Özsoyoğlu, A. Al-Hamdani (Case Western Reserve University).

Funding Sources: A joint grant of Scientific and Technical Research Council of Turkey (TÜBİTAK) under grant number 100U024 and National Science Foundation of the USA under grant number INT-9912229.

3.1 Metadata-Based Modeling of the Web

A recent approach to increase the quality of Web search is associating metadata to the resources on the Web. To this end, there are various standardization efforts and initiatives, such as the Dublin Core Framework, RDF, Semantic Web and Topic Maps. In [9, 10], we address the problem of modeling Web information resources using expert knowledge and personalized user information for improved Web searching capabilities. We propose a "Web information space" model, which is composed of Web-based information resources (HTML/XML documents on the Web), expert advice repositories (domain-expert-specified metadata for information resources), and personalized information about users (captured as user profiles that indicate users' preferences about experts as well as users' knowledge about topics).

Expert advice, the heart of the Web information space model, is specified using topics and relationships among topics (called metalinks), along the lines of the recently proposed topic maps. Topics and metalinks constitute metadata that describe the contents of the underlying HTML/XML Web resources. The metadata specification process is semi-automated. It exploits XML DTDs to allow domain-expert guided mapping of DTD elements to topics and metalinks. In particular, the domain expert specifies a mapping between the entities of our metadata model (topics and metalinks) and the XML DTDs in the corresponding domain(s). An agent then traverses the Web, extracts topics and metalinks for those XML files conformant with the input DTD, and stores them into a local object-relational database management system, which will then serve as an expert advice (metadata) repository for these visited Web resources.

To demonstrate the practicality and usability of the proposed Web information space model, we have created a prototype expert

advice repository of more than one million topics and metalinks for DBLP (Database and Logic Programming) Bibliography data set.

3.2 Querying Web Metadata: Native Score Management and Text Support in Databases

In this work, we discuss the issues involved in adding a native score management system to object-relational databases, to be used in querying web metadata (that describes the semantic content of web resources) [11, 12]. As described in the preceding section, the web metadata model is based on topics (representing entities), relationships among topics (metalinks), and importance scores (sideway values) of topics and metalinks. We extend database relations with scoring functions and importance scores. We add to SQL score-management clauses with well-defined semantics, and propose the sideway-value algebra (SVA), to evaluate the extended SQL queries.

SQL extensions include clauses for propagating input tuple importance scores to output tuples during query processing, clauses that specify query stopping conditions, threshold predicates—a type of approximate similarity predicates for text comparisons, and user-defined-function-based predicates. The propagated importance scores are then used to rank and return a small number of output tuples. The query stopping conditions are propagated to SVA operators during query processing. We show that our SQL extensions are well-defined, meaning that, given a database and a query Q, the output tuples of Q and their importance scores stay the same under any query processing scheme.

3.3 Focused Web Crawling

A preliminary step for extracting and querying metadata for Web resources is gathering all and only the relevant Web resources for a particular application domain or topic of interest. To this end, in [13] we propose a *rule-based focused crawling* strategy to crawl the Web and construct a repository of relevant Web pages. A focused crawler is an agent that concentrates on a particular target topic, and tries to visit and gather only relevant pages from the Web. In the literature, one of the approaches for focused crawling is using a canonical topic taxonomy and a text classifier to train the crawler so that those URLs that most probably point to on-topic pages will be identified and prioritized. Our research explores using simple rules derived from the linkage statistics among the topics of a taxonomy while deciding on the crawler's next move, i.e., to select the URL to be visited next. The rule based approach improves the harvest rate and coverage of the taxonomy-based focused crawler and also enhances it to support *tunneling*. More specifically, the rule based crawler can follow a path of off-topic pages that may at last lead to high quality on-topic pages. More information on our projects in *Web Databases* can be found through http://www.cs.bilkent.edu.tr/~bilweb.

References

[9] I. S. Altingovde, S. A. Özel, Ö. Ulusoy, G. Özsoyoğlu, Z. M. Özsoyoğlu. Topic-Centric Querying of Web Information Resources. In *Proceedings of the Database and Expert Systems Applications (DEXA'01)*, *Lecture Notes in Computer Science (Springer Verlag)*, vol.2113, (Munich, Germany, September 2001) 699-711.

[10] S. A. Özel, I. S. Altingovde, Ö. Ulusoy, G. Özsoyoğlu, Z. M. Ösoyoglu. Metadata-Based Modeling of Information Resources on the Web. *Journal of the American Society for Information Science and Technology (JASIST)*, 55, 2 (January 2004), 97-110.

[11] G. Özsoyoğlu, Abdullah Al-Hamdani, I. S. Altingovde, S. A. Özel, Ö. Ulusoy, Z. M. Özsoyoğlu. Sideway Value Algebra for Object-Relational Databases. In *Proceedings of the International Conference on Very Large Databases (VLDB'02)*. (Hong Kong, August 2002), 59-70.

[12] G. Özsoyoğlu, I. S. Altingovde, A. Al-Hamdani, S. A. Özel, Ö. Ulusoy, Z. M. Özsoyoğlu. Querying Web Metadata: Native Score Management and Text Support in Databases. *ACM Transactions on Database Systems*, 29, 4 (December 2004), 581-634.

[13] I. S. Altingovde, Ö. Ulusoy. Exploiting Interclass Rules for Focused Crawling**.** *IEEE Intelligent Systems*, 19, 6 (November-December 2004), 66-73.

4. Mobile Computing

Researchers: Ö. Ulusoy, M. Karakaya, İ. Körpeoğlu, S. Çıracı.
Past Researchers: , Y. Saygın, E. Kayan, J. Tayeb, G. Gök, I. Yoncacı, G. Yavaş.
Collaborators: O. Wolfson (University of Illinois at Chicago), K. Y. Lam (City University of Hong Kong), D. Katsaros, Y. Manolopoulos (Aristotle University), A. K. Elmagarmid (Purdue University).

Funding Sources: Scientific and Technical Research Council of Turkey (TÜBİTAK) under grant number EEEAG-246, NATO Collaborative Research Program under grant number CRG 960648, and the bilateral program of scientific cooperation between Turkey and Greece (TÜBİTAK grant number 102E021 and *Γ.Γ.Ε.Τ.*).

4.1 Moving Object Processing

Our earlier work on moving object processing includes indexing locations of moving objects. In [14], we propose an indexing technique for moving objects based on a variant of the quadtree data structure in which the indexing directory is in primary memory and the indexed data resides in secondary storage. The method is useful in any application that involves dynamic attributes whose values vary continuously according to a given function of time. Our approach is based on the key idea of using a linear function of time for each dynamic attribute that allows us to predict its value in the future. We contribute an algorithm for regenerating the quadtree-based index periodically to minimize CPU and disk access cost.

Another important issue in moving object database management is to provide support for processing location-dependent queries, where the answer to a query depends on the current location of

the user who issued the query. A location-dependent query can become more difficult to process when it is submitted as a continuous query for which the answer changes as the user moves. In [15], we present an efficient method to monitor the locations of moving objects so that timely and accurate results can be returned to location dependent continuous queries, while minimizing the location update cost. Location-dependent queries from mobile clients may also be associated with timing constraints on their response times. In [16], we propose a method to monitor the locations of moving users/objects based on the real-time criticality of location dependent continuous queries, so that higher levels of accuracy can be achieved for the results returned to the queries categorized with higher criticality. Another related issue in processing location dependent continuous queries is the transmission of query results to the users. In [17], various methods that can be used to determine the transmission time of query results are investigated with the goal of minimizing data transmission costs.

In [18], we present a data mining algorithm for the prediction of user movements in a mobile computing system. The algorithm proposed is based on mining the mobility patterns of users, forming mobility rules from these patterns, and finally predicting a mobile user's future movements by using these rules.

4.2 Mobile Data Management

Dissemination of data by broadcasting may induce high access latency in case the number of broadcast data items is large. In [19], we propose two methods to reduce client access latency for broadcast data. Our methods are based on analyzing the broadcast history (i.e., the chronological sequence of items that have been requested by clients) using data mining techniques. The proposed methods are implemented on a Web log, and it is shown that the proposed rule-based methods are effective in improving the system performance in terms of average latency as well as cache hit ratio of mobile clients. Our other work on broadcast scheduling considers a pull-based data delivery environment [20]. We propose a variant of the *Longest Wait First* heuristic in scheduling data broadcast, which provides a practical implementation of the heuristic by avoiding the decision overhead.

One of the features that a mobile computer should provide is disconnected operation, which is performed by hoarding. The process of hoarding can be described as loading the data items needed in the future to the client cache prior to disconnection. Automated hoarding is the process of predicting the hoard set without any user intervention. In [21], we describe a generic, application-independent technique for determining what should be hoarded prior to disconnection. Our method utilizes association rules that are extracted by data mining techniques for determining the set of items that should be hoarded to a mobile computer prior to disconnection.

While there has been much research interest in mobile computing issues, an issue that has not received much attention is the management of the database of a mobile system under timing constraints. In [22], we present a mobile database system model that takes into account the timing requirements of applications supported by mobile computing systems. We provide a transaction execution model with alternative execution strategies for mobile transactions and evaluate the performance of the system under various mobile system characteristics, such as the number of mobile hosts in the system, the handoff process, disconnection, coordinator site relocation, and wireless link failure.

References

[14] J. Tayeb, Ö. Ulusoy, O. Wolfson, A Quadtree Based Dynamic Attribute Indexing Method**.** *The Computer Journal*, 41, 3 (1998) 185-200.

[15] K. Y. Lam, Ö. Ulusoy, et al., An Efficient Method for Generating Location Updates for Processing of Location-Dependent Continuous Queries**.** In *Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA'01)*. (Hong Kong, April 2001) 218-225.

[16] Ö. Ulusoy, I. Yoncacı, K. Y. Lam. Evaluation of a Criticality-Based Method for Generating Location Updates. In *Proceedings of Workshop on Database Mechanisms for Mobile Applications, Lecture Notes in Informatics*, vol.43, (Karlsruhe, Germany, April 2003) 94-105.

[17] G. Gök, Ö. Ulusoy. Transmission of Continuous Query Results in Mobile Computing Systems. *Information Sciences*, 125, 1-4 (2000) 37-63.

[18] G. Yavaş, D. Katsaros, Ö. Ulusoy, Y. Manolopoulos. A Data Mining Approach for Location Prediction in Mobile Environment. *Data and Knowledge Engineering*, 54, 2 (2005) 121-146.

[19] Y. Saygın, Ö. Ulusoy. Exploiting Data Mining Techniques for Broadcasting Data in Mobile Computing Environments. *IEEE Transactions on Knowledge and Data Engineering*, 14, 6 (November/December 2002) 1387-1399.

[20] M. Karakaya, Ö. Ulusoy. Evaluation of a Broadcast Scheduling Algorithm. In *Proceedings of Advances in Databases and Information Systems (ADBIS'01), Lecture Notes in Computer Science (Springer Verlag)*, vol.2151, (Vilnius, Lithuania, September 2001) 182-195.

[21] Y. Saygın, Ö. Ulusoy, A. K. Elmagarmid. Association Rules for Supporting Hoarding in Mobile Computing Environments. In *Proceedings of IEEE International Workshop on Research Issues on Data Engineering (RIDE'00)*, (San Diego, CA, USA, February 2000) 71-78.

[22] E. Kayan, Ö. Ulusoy. An Evaluation of Real-Time Transaction Management Issues in Mobile Database Systems. *The Computer Journal,* 42, 6 (November 1999) 501-510.

Data Management Research at the Middle East Technical University

Nihan K. Cicekli, Ahmet Cosar, Asuman Dogac, Faruk Polat, Pinar Senkul, I. Hakki Toroslu and Adnan Yazici

Department of Computer Engineering Database Group
Middle East Technical University (METU)
06531 Ankara Turkey

1. INTRODUCTION

The Middle East Technical University (METU) (http://-www.metu.edu.tr) is the leading technical university in Turkey. The department of Computer Engineering (http://-www.ceng.metu.edu.tr) has twenty seven faculty members with PhDs, 550 undergraduate students and 165 graduate students. The major research funding sources include the Scientific and Technical Research Council of Turkey (TÜBİTAK), the European Commission, and the internal research funds of METU. Data management research conducted in the department is summarized in this article.

2. WEB SERVICES AND SEMANTIC WEB TECHNOLOGIES

The research in the semantic Web services area has concentrated upon the application of this technology in two important sectors: healthcare through the Artemis project (http://www.srdc.metu.edu.tr/webpage/projects/-artemis/) and tourism through the Satine project (http://-www.srdc.metu.edu.tr/webpage/projects/satine/).

2.1 The Artemis Project

The Artemis project provides the interoperability of medical information systems through semantically enriched Web services. An essential element in defining the semantic of Web services is domain knowledge. Medical informatics is one of the few domains to have considerable domain knowledge exposed through standards as mentioned above. These standards offer significant value in terms of expressing the semantic of Web services in the healthcare domain. In the Artemis project, prominent healthcare standards are used to semantically annotate Web services as follows:

- HL7 has categorized the events in the healthcare domain by considering service functionality that reflects the business logic in this domain. We use this classification as a basis for defining the service action semantics through a "Service Functionality Ontology". In this way, semantic discovery of Web services is facilitated.

- Given the complexity of clinical domain, the Web service messages exchanged have innumerous segments of different types and optionality. To make any use of these messages at the receiving end, their semantics must be clearly defined. We annotate the Web services through the reference information models of Electronic Healthcare Record (EHR) standards.

The details of this work is presented in the following publication:

- Dogac, A., Laleci, G., Kirbas, S., Kabak, Y., Sinir, S., Yildiz, A., Gurcan, Y., "Artemis: Deploying Semantically Enriched Web Services in the Healthcare Domain", Information Systems Journal, Elsevier, to appear.

Using archetypes is a promising approach for providing semantic interoperability among healthcare systems. To realize archetype based interoperability, healthcare systems need to discover the existing archetypes based on their semantics, annotate their archetypes with ontologies, compose templates from archetypes and retrieve corresponding data from the underlying medical information systems. In the Artemis project, we use ebXML Registry semantic constructs for annotating, storing, discovering and retrieving archetypes.

The details of this work is presented in the following publication:

- Dogac, A., Laleci, G. B., Kabak, Y., Unal, S., Beale, T., Heard, S., Elkin, P., Najmi, F., Mattocks, C., Webber, D., "Exploiting ebXML Registry Semantic Constructs for Handling Archetype Metadata in Healthcare Informatics", Intl. Journal of Metadata, Semantics and Ontologies, to appear.

In the Artemis project, AMEF (Artemis Message Exchange Framework) is developed to provide the exchange of meaningful clinical information among healthcare institutes through semantic mediation. The framework involves first providing the mapping of source ontology into target message ontology with the help of a mapping tool that produces a mapping definition. This mapping definition is then used to automatically transform the source ontology message instances into target message instances. Through a prototype implementation, we demonstrate how to mediate between HL7 Version 2 and HL7 Version 3 messages. However, the framework proposed is generic enough to mediate between any incompatible healthcare standards that are currently in

use. The details of this work is presented in the following publication:

- Bicer, V., Laleci, G., Dogac, A., Kabak, Y., "Artemis Message Exchange Framework: Semantic Interoperability of Exchanged Messages in the Healthcare Domain", ACM Sigmod Record, Vol. 34, No. 3, September 2005.

2.2 The SATINE Project

The SATINE project addresses the interoperability in the travel domain. The tourism industry today is the second largest economic sector, after manufacturing in the world. Tourism embarked on eBusiness earlier than other sectors. Currently, travel information services are dominantly provided by Global Distribution Systems (GDSs). All the airlines, many hotel chains and car rental companies list their inventory with major GDSs. A GDS gives its subscribers pricing and availability information for multiple travel products such as flights. Travel agents, corporate travel departments, and even Internet travel services, subscribe to one or more GDSs. However, small and medium-sized enterprises, for example "bed and breakfast" type accommodation or companies hiring bicycles, restaurants and a host of others cannot participate to GDS-based eBusiness activities because selling their products through GDSs is too expensive for them.

Furthermore, GDSs are legacy systems and suffer from a number of problems: they mostly rely on private networks, are mainly for human use, have difficult to use cryptic interfaces, have limited speed and search capabilities, and are difficult to interoperate with other systems and data sources. The implication is that the tour operators, travel agencies, etc. cannot benefit fully from the advantages of electronic business-to-business trading.

In order to facilitate eBusiness, the travel industry has formed a consortium called the Open Travel Alliance (OTA). OTA produces XML schemas of message specifications to be exchanged between the trading partners, including availability checking, booking, rental, reservation, query services, and insurance. However, not every travel company's applications can be expected to produce and consume OTA compliant messages.

In the SATINE project, we describe how to deploy semantically enriched travel Web services and how to exploit semantics through Web service registries to addres the problems mentioned. We also address the need to use the semantics in discovering both Web services and Web service registries through peer-to-peer technologies. The mechanisms are described in detail in the following publication:

- Dogac, A., Kabak, Y., Laleci, G., Sinir, S., Yildiz, A., Kirbas, S., Gurcan, Y., "Semantically Enriched Web Services for the Travel Industry", ACM Sigmod Record, Vol. 33, No. 3, September 2004.

Web services, similar to their real life counterparts, have several properties and, thus, truly useful semantic information can only be defined through standard ontology languages. In the SATINE project, mechanisms to enrich ebXML registries through OWL-S ontologies for describing the Web service semantics are developed. Particularly, how the various constructs of OWL can be mapped to ebXML classification hierarchies and how the services are discovered through standardized queries by using the ebXML query facility are described.

Detailed information on these mechanisms is available in the following publication:

- Dogac, A., Kabak, Y., Laleci, G. B., Mattocks, C., Najmi, F., Pollock, J., "Enhancing ebXML Registries to Make them OWL Aware", Distributed and Parallel Databases Journal, Springer Verlag, Vol. 18, No. 1, July 2005, pp. 9-36.

Finally, how privacy issues can be handled semantically in Web services is addressed in the following publication:

- Tumer, A., Dogac, A., Toroslu, I. H., "A Semantic based Privacy Framework for Web Services", WWW'03 workshop on E-Services and the Semantic Web (ESSW 03), Budapest, Hungary, May 2003.

3. VIDEO DATABASES, SPATIO-TEMPORAL DATABASES

Content-based querying of multimedia data is a relatively new subject, which has arisen fast with the improvements in data processing and communication systems technologies. We present a spatio-temporal video data model that allows efficient and effective representation and querying of spatio-temporal objects in videos. The data model is focused on the semantic content of video streams. Objects, events and activities performed by objects, and temporal and spatial properties of objects are the main interests of the model. Spatial and temporal relationships between objects and events are dynamically calculated with the query processing methods introduced in this paper. The model is flexible enough to define new relationship types between objects without changing the data model and supports fuzzy querying of spatio-temporal objects in videos. Index structures are used for effective storage and retrieval of the mentioned properties. A prototype of the proposed model has been implemented. The prototype allows various spatio-temporal queries along with some fuzzy ones, and it is capable of implementing many other queries without any major changes in the data model.

- Koprulu, M., Cicekli, N. K., Yazici, A., "Spatio-temporal Querying in Video Databases", Information Sciences, Vol. 160, 2004, pp. 131-152.

- Koprulu, M., Cicekli, N. K., and Yazici, A., "Spatio-Temporal Querying in Video Databases", Proc. of the Sixth International Conf. on Flexible Query Answering Systems (FQAS'2002), Denmark, Oct 2002.

- Yazici, A., Yavuz, O., and George, R., "An MPEG-7 Based Video Database management System, Flexible Querying and Reasoning in Spatio-Temporal Databases: Theory and Applications", In Springer's Geo-sciences/Geoinformation series by Springer Verlag, 2004, pp. 181-210.

Depending on the proposed content-based spatio-temporal video data model, a natural language interface is implemented to query the video data. The queries, which are given as English sentences, are parsed using Link Parser, and the semantic representations of given queries are extracted from their syntactic structures using information extraction techniques. At the last step, the extracted semantic

representations are used to invoke the related parts of the underlying spatio-temporal video data model to retrieve the results of the queries.

- Erozel, G., Cicekli, N. K., Cicekli, I., "Natural Language Interface on a Video Data Model", Proceedings of IASTED DBA 2005, Austria.

4. INTELLIGENT DATABASE SYSTEMS, FUZZY LOGIC AND DATABASE MODELING

Next generation information system applications require powerful and intelligent information management that necessitates an efficient interaction between database and knowledge base technologies. It is also important for these applications to incorporate uncertainty in data objects, integrity constraints, and applications.

Fuzzy relational database models generalize the classical relational database model by allowing uncertain and imprecise information to be represented and manipulated. We introduce fuzzy extensions to the relational database model. Within this framework of fuzzy data representation, similarity, conformance of tuples, the concept of fuzzy functional dependencies, and partial fuzzy functional dependencies are utilized to define the fuzzy key notion, transitive closures, and fuzzy normal forms. The fuzzy object-oriented data model is a fuzzy logic-based extension to the object-oriented database model, which permits uncertain data to be explicitly represented. The details are presented in the following publications:

- Koyuncu, M., and Yazici, A., "A Fuzzy Knowledge-Based System for Intelligent Retrieval", IEEE Transactions on Fuzzy Systems, to appear.
- Sozer, A., and Yazici, A., "Design and Implementation of Index structures for Fuzzy Spatial Databases", International Journal of Intelligent Systems, to appear.
- Bahar, O., and Yazici, A., "Normalization And Lossless Join Decomposition of Similarity-Based Fuzzy Relational Databases", International Journal of Intelligent Systems, Vol. 19, No. 10, October 2004, pp. 885-918.
- Aygun, R. S., and Yazici, A., "Modeling and Management of Fuzzy Information in Multimedia Database Applications", Multimedia Tools and Applications, Vol. 24, No.1, September 2004, pp. 29-56.
- Koyuncu, M., and Yazici, A., "IFOOD: An Intelligent Fuzzy Object-Oriented Database Architecture", IEEE Trans. on Knowledge and Data Engineering, September 2003, pp. 1137-1154.
- Sozat, M. I., Yazici, A., "A Complete Axiomatization for Fuzzy Functional and Multivalued Dependencies in Fuzzy Database Relations", Fuzzy Sets and Systems, Vol. 117/2, 2001, pp. 161-181
- Yazici, A., Zhu, Q., Sun, N., "Semantic Data Modeling of Spatiotemporal Database Applications", Int. Journal of Intell. Systems, Vol. 16, No. 7, July 2001, pp. 881-904

5. WORKFLOWS

The research on workflow management systems has concentrated on modeling and scheduling under resource allocation systems. In addition to the temporal constraints corresponding to the order of tasks, constraints related to resource allocation are also equally important. Workflow scheduling should include the decisions about which resources are allocated to which tasks, parallel to the task ordering decision. To solve this, two approaches have been studied. In the first one, we proposed an architecture to specify and schedule workflows under resource allocation constraints as well as temporal and causality constraints:

- Senkul, P., and Toroslu, I. H., "An architecture for workflow scheduling under resource allocation constraints", Information Systems, Vol. 30, Issue 5, July 2005, pp. 399-422.

In the second approach, we developed a new logical formalism, called Concurrent Constraint Transaction Logic (CCTR), which integrates Constraint Logic Programming (CLP) and Concurrent Transaction Logic, and a logic-based workflow scheduler that is based on this new formalism. The semantics of the CCTR modeling of a workflow represent a schedule that contains both an execution ordering that the specified workflow can execute, and a set of resource assignments to the tasks of the workflow satisfying the given constraints. The details of this work is presented in the following publication:

- Senkul, P., Kifer, M., Toroslu, I. H., "A Logical Framework for Scheduling Workflows under Resource Allocation Constraints", VLDB 2002, pp. 694-705.

As another research direction, we have proposed a logic-based framework for the specification and execution of workflows. The proposed approach is based on the Kowalski and Sergot's Event Calculus:

- Cicekli, N. K., and Cicekli, I., "Formalizing the Specification and Execution of Workflows using the Event Calculus", Information Sciences, to appear.

6. DATA MINING

Most research on data mining has focused on processing single relations. Recently, however, multi-relational data mining has started to attract interest. One of the earliest works on multi-relational data mining is the query flocks technique, which extends the concept of traditional association rule mining with a "generate-and-test" model for different kinds of patterns. One possible extension of the query flocks technique is the addition of view definitions including recursive views. Although in our system the query flock technique can be applied to a database schema including both the Intensional Database (IDB) or rules, and the Extensible Database (EDB) or tabled relations, we have designed an architecture to compile query flocks from datalog into SQL in order to be able to use commercially available Database Management Systems (DBMS) as an underlying engine of our system. Currently, we are extending our work on multi-relational data mining using inductive logic programming for discovering rules. This work is presented in the following publication:

- Toroslu, I. H., Yetisgen-Yildiz, M., "Data Mining in Deductive Databases Using Query Flocks", Expert Systems with Applications, Vol. 28, Issue 3, April 2005, pp. 395-407.

One of the most well-known data mining problems is the sequential pattern mining. The main drawback of sequential pattern mining is the large number of sequential patterns discovered, which makes it harder for the decision maker to interpret them. Thus, a new parameter is added to the definition of the sequential pattern mining problem that represents the repetitions of the sequences. In the following publication, this problem is introduced together with its possible applications and advantages over ordinary sequential pattern mining:

- Toroslu, I. H., "Repetition Support and Mining Cyclic Patterns", Expert Systems with Applications, Vol. 25, issue 3, october 2003, pp. 303-311.

7. QUERY PROCESSING

The "Multiple Query Optimization" (MQO) problem, a well-known query-processing problem in databases, has been studied in the database literature since the 1980s. MQO tries to reduce the execution cost of a group of queries by performing common tasks only once. In our work, we assume that, at the beginning of the optimization, all promising alternative plans have been generated and shared tasks are identified. Our algorithm finds an optimal solution to the MQO problem. This form of MQO problem has been formulated as an NP-complete optimization problem where several heuristic functions are used to guide an A* search. In the following work, we propose the first dynamic programming approach to this problem:

- Toroslu, I. H., and Cosar, A., "Dynamic Programming Solution for Multiple Query Optimization Problem", Information Processing Letters, Vol. 92, Issue 3, 15 November 2004, pp. 149-155.

There has been several other previous works done by the members of the database group at METU focused on other aspects of multiple query optimization and semantic query optimization. Our most recent work on semantic query optimization is as follows:

- Polat, F., Cosar, A., Alhajj, R., "The Semantic Information-based Alternative Plan Generation for Multiple Query Optimization", Information Sciences, Elsevier, Vol. 137/1-4, 2001, pp. 103-133.

8. OBJECT-ORIENTED DATABASES

Research on object-oriented databases is primarily based on extensibility and dynamic schema evolution. We have benefited from having an object algebra maintaining closure that makes it possible to have output from a query persistent in the hierarchy. We automate the process of properly placing new classes in the class hierarchy. We are able to map a view, which can easily be defined in our model, that is intended to be persistent into a class. The object algebra is utilized to handle basic schema evolution functions without requiring any special set of built-in functions.

- Alhajj, R., Polat, F., and Yilmaz, C., "Views as First-Class Citizens in Object-Oriented Databases", *The VLDB Journal*, Vol. 14, No. 2, April 2005, pp. 155-169.

- Alhajj, R., and Polat, F., "Rule-Based Schema Evolution in Object-Oriented Databases", *Knowledge-Based Systems*, Vol. 16, No. 1, Jan. 2003, pp. 47-57.

- Alhajj, R., and Polat, F., "Reengineering of Relational Databases into Object-Oriented: Constructing the Class Hierarchy and Migrating the Data", *Proceedings of the IEEE Working Conference on Data Reverse Engineering WCRE 2001*, IEEE Press, Stuttgart, Germany, Oct. 2001, pp. 335-344.

- Alhajj, R., and Polat, F., "Transferring Database Contents from a Conventional Information System to a Corresponding Existing Object Oriented Information System", *Proceedings of the ACM Annual Symposium on Applied Computing*, ACM Press, Las Vegas, USA, Mar. 2001, pp. 220-224.

9. BIOINFORMATICS

We worked on the problem of identifying Differentially Expressed Genes (DEG) and improved the power of PaGE by estimating prior probability used in the confidence computation. We developed two methods based on the *q-values* and *maximum likehood* approaches to find DEG on a dataset of microarray experiments for pattern generation. We formulate the control problem for dynamic discrete regulatory networks and defined various control-monitor strategies. The Multiobjective control problem is further studied and a case study is done for proof of correctness.

- Abul, O., Alhajj, R., Polat, F., and Barker, K., "Finding Differentially Expressed Genes: Pattern Generation", *Bioinformatics*, Vol. 21, No. 4, Feb 2005, pp. 445-450.

- Abul, O., Alhajj, R., and Polat, F., "Importance of Monitoring in Developing Optimal Control Policies for Probabilistic Boolean Genetic Networks", *Information Processing and Management of Uncertainty in Knowledge-Based Systems* (IPMU2004), Perugia, Italy, July 2004, pp. 1145-1152.

- Abul, O., Alhajj, R., and Polat, F., "Markov Decision Processes Based Optimal Control Policies for Probabilistic Boolean Network", *Proc. of IEEE Symposium on Bioinformatics and Bioengineering* (BIBE2004), Taichung, Taiwan, May 19-21, 2004, pp. 337-344.

- Abul, O., Alhajj, R., Polat, F., and Barker, K., "Finding Differentially Expressed Genes: Pattern Generation using q-values", *Proceedings of the ACM Annual Symposium on Applied Computing (SAC)*, Cyprus, Mar. 2004, pp. 138-142.

- Abul, O., Lo, A., Alhajj, R., Polat, F., and Barker, K., "Cluster Validity Analysis Using Subsampling", *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 2, Washington DC, Oct. 2003, pp. 1434-1440.

Report on the Workshop on Wrapper Techniques for Legacy Data Systems

Ph. Thiran[1,2], T. Risch[3], C. Costilla[4], J. Henrard[1], Th. Kabisch[5], J. Petrini[3], W-J. van den Heuvel[6], J-L. Hainaut[1]

[1]University of Namur, Belgium

[2]Eindhoven University of Technology, The Netherlands

[3]Uppsala University, Sweden

[4]TU Madrid, Spain

[5]TU Berlin, Germany

[6]Tilburg University, The Netherlands

This report summarizes the presentations and discussions of the first workshop on Wrapper Techniques for Legacy Data Systems which was held in Delft on November 12 2004. This workshop was co-located with the 2004 WCRE conference. This workshop entails to our best knowledge the first in its kind, concentrating on challenging research issues regarding the development of wrappers for legacy data systems.

1. INTRODUCTION

Research and development in legacy data wrapping are of paramount importance given the fact that the combination of distributed Internet technology and wrappers opens up new opportunities to unlock the business value that is stored in legacy systems. In fact, legacy systems hold services that remain useful beyond the means of the technology in which they were originally implemented. Wrappers are used to facilitate the reuse of key portions of the legacy systems for their integration into novel business systems, like web applications.

Grossly speaking, a wrapper can be perceived as a model converter, i.e., a software component that translates data and queries from the interface (model and languages) of a legacy data system to another, abstract interface intended to client applications and users. A critical issue for the development of wrappers is that legacy data systems vary widely in their support for data manipulation and description. Moreover, the data schema of the underlying data source can be unavailable or incomplete. This is typically the case for web sources. In these cases, the data schema should preferably be automatically derived from the actual data that is captured in the (web) resource.

After a short introduction, the day started with a technical session of papers, followed by a plenary discussion about some issues in data wrapping. In the following, a summary of the main points of each of these is presented.

2. TECHNICAL PRESENTATIONS

After a blind review process, four papers were selected for presentation and publication. These workshop proceedings are published on-line by the university press of the Eindhoven University of Technology and are accessible from the workshop website, which will remain open at http://wwwis.win.tue.nl/wrapper04/.

Göldner et al. [1] deal with the maintenance of Web wrappers. They introduce an approach which uses caching of sample data and structures in combination with content recognition to enhance robustness of wrappers and to automate the wrapper maintenance process. Petrini and Risch [2] describe a query translating wrapper of relational databases from RDF-based semantic web queries into SQL to enable access to relational databases from semantic web tools. The particular problem is here efficient dynamic optimization of queries to the wrapper. In their paper, Vila and Costilla [3] present a mediator-wrapper architecture based on an extractor data model placed between each Web data source and its wrapper linked to a specific ontology. Finally, Jean Henrard and al. [4] address the difficult problem of migrating application programs from a legacy data manager, such as a COBOL file system, to a modern DBMS, such as a relational database management system. The approach relies on the concept of *inverse wrappers*; that is, wrappers that simulate the legacy API on top of the new database.

3. DISCUSSION

The workshop was concluded by a plenary discussion in order to provide a definition of wrappers as well as to summarize the state of the art in this research field. In this section, we will briefly sketch the main highlights of the discussions, focusing on wrapper definition and on the evolution of modeling languages.

3.1 Wrapper Definition

The purpose of a wrapper is to make one or several data sources queryable in terms of an agreed-upon query language.

Individual wrappers can be defined for each source, e.g., for an existing on-line legacy system, an existing data production system, etc. To improve code reuse, generic wrappers can be defined for a class of data sources, e.g., for all relational databases providing JDBC drivers and SQL queries, or for HTML interfaces. Generic wrappers permit easy inclusion of new sources into a mediator system. By object-orientation one can further refine this code sharing by providing a hierarchy of generic wrappers, e.g., DB2 wrappers are specializations of general relational database wrappers, etc.

Wrapper can provide several types of query languages for data sources, ranging from sophisticated query languages to low-level data access interfaces. For example, XML or HTML documents on the web, or legacy systems, are not accessed through a query language but rather through a data access interface. The wrapper must then provide enough query processing capabilities. For example, [1] describes a *query enabling* wrapper for web interfaces based on Xpath. By contrast, if the wrapped XML documents are managed by a system providing SQL interfaces, the wrapper is *query translating* and mainly concerned with translating and decomposing queries of the common query language used by the mediator systems into queries of the wrapped data source. Query translating wrappers can be complex, since a data source with a query language such as SQL provides opportunities for advanced optimization choices to split work between the wrapper and the back-end query processor, while a primitive non-query based data source interface provides fewer choices for optimization [2].

Another important aspect of a wrapper/mediator system is how it is accessed from other systems. For example, SQL provides a well established standard for data access and a mediator system providing standardized SQL interfaces, such as IBM DB2 Information Integrator, can be used by many applications. When legacy systems that use an outdated interfaces, such as COBOL file access, are to be left unchanged when upgrading to, say, relational database technology, a component that wraps the relational databases needs to be linked to the legacy application. This can be done either by replacing the native COBOL API with the wrapper (as with Microfocus COBOL), in which case the programs are left unaltered, or by replacing the native I/O statements with wrapper invocations, in which case the programs have to be modified in a simple way that is particularly easy to automate [4].

3.2 Evolution of the Models and Languages

Over the years, several models have been used for describing wrapped data, such as the relational model which has been frequently used. Since XML has become the standard information exchange language through the Web, there has been a shift to using it as the pivot modeling language which the others have to be translated to or from [3]. The recent trends are to use RDF [2]. This shift has been motivated by the need for high level semantic descriptions of web information for use in, e.g, reasoning tools. The original use of XML was for tree-based structuring of individual text documents. For these documents DTDs provide a primitive schema definition language. For more data-oriented use of XML, XML Schema provides a sophisticated XML-based type and structure system.

By contrast, the data model of RDF is based on triples, <subject, predicate, object>, that can annotate any web resource with arbitrary properties referencing other web resources. The RDF model can be seen as a binary relational data model with well-defined semantics based on logic. An RDF database forms a graph of associations between different web resources rather than an XML-markup that describes a single document. The languages built on top of RDF, e.g., RDF-Schema and OWL also are formally based on logic, which makes them more appealing than, e.g., XML Schema, for describing knowledge. This makes XML and its derivatives suited for data integration and high level reasoning.

4. TOPICS FOR FUTURE WORK

The observations drawn at the workshop and our understanding of the whole area of wrapping yield a number of open questions to be discussed in the future, possibly at the next WRAP workshop. Open issues include:

- Both data and web wrappers may be leveraged using Semantic Web technologies (e.g., Thesaurus or Ontology) in order to infuse more context knowledge in the wrapper.
- Regardless of the type of wrapper, an important issue that needs to be resolved is how these wrappers may be developed or generated in an effective manner for different kinds of data sources so that specific wrapper interfaces can be generated for specific data sources. A critical success factor for developing wrappers seems to be to (semi-) automatically acquire deep generic understanding about each kind of mapping between the structure of the data sources and that of the data advertised by the wrapper.
- Another research direction could be directed at making wrappers more robust and enhance the automation degree; this might be based on the integration of statistical approaches/automated learning approaches in order to make the wrapper framework more "intelligent". It seems especially important to overcome difficulties if the structure of data (either provided on the web or in some database/file system) is not entirely known a priori.
- Moreover, the evolution of wrappers, once in place, will most likely become an important issue of future research.
- Lastly, the issue of which functionalities in a Mediator-Wrapper system can be delegated to the wrapper will be an interesting topic. E.g., some approaches rely on a wrapper-supported query planning.

REFERENCES

[1] Ch. Göldner, Th. Kabisch and J. G. Süß, Developing robust wrapper-systems with Content Based Recognition, in *WRAP*, 2004.

[2] J. Petrini and T. Risch, Processing Queries over RDF views of Wrapped Relational Databases, in *WRAP*, 2004.

[3] J. Vila and C. Costilla, Heterogeneous Data Extraction in XML, in *WRAP*, 2004.

[4] J. Henrard, A. Cleve and J.-L. Hainaut, Inverse Wrappers for Legacy Information Systems Migration, in *WRAP*, 2004.

Exchange, Integration, and Consistency of Data. Report on the ARISE/NISR Workshop.

Leopoldo Bertossi (Carleton University), **Jan Chomicki** (University at Buffalo),
Parke Godfrey (York University), **Phokion G. Kolaitis** (IBM Almaden Research Center),
Alex Thomo (University of Victoria), and **Calisto Zuzarte** (IBM CAS, Toronto Lab.)

1 Introduction

The "ARISE/NISR Workshop on Exchange and Integration of Data" was held at the IBM Center for Advanced Studies, Toronto Lab., between October 7-9, 2004.

The Advanced Research Initiative for Software Excellence (ARISE), now referred to as the National Institute for Software Research (NISR), was founded by the National Research Council's Institute for Information Technology, the IBM Toronto Lab., the University of Toronto, the University of Waterloo, and York University, as an effort to create a national institute that will enhance knowledge in the area of software, helping Canada to increase its competitive advantage in the field. NISR's program includes the creation and support of workshops, seminars, courses, and research projects. This first ARISE workshop was organized by Leopoldo Bertossi, Parke Godfrey, Paul Smith (IBM Toronto Lab.), and Calisto Zuzarte; it congregated a large number of researchers. Details and presentations are available at the workshop web site: http://www.scs.carleton.ca/∼diis/arise/workshop.html.

2 Workshop Contents

Kelly Lyons (IBM Toronto Lab. and technical steering committee of NISR) provided the opening remarks. The three keynote presentations, "INFOMIX: Data Integration meets Nonmonotonic Deductive Databases" by Thomas Eiter, "Model Management: Generic Operators for Schema Mappings and Database Integration" by Philip A. Bernstein, and "Hyper: A Framework for P2P Information Integration" by Maurizio Lenzerini, represented well the three areas of research that emerged during the workshop: (a) Virtual Data Integration; (b) Data Exchange and Schema Mappings; and (c) Inconsistency Handling in Databases. It was particularly interesting to see the connections between them. Data exchange and mediator-based data integration share several ideas, concepts, and techniques, but the corresponding communities have stayed relatively distant from each other. Consistency issues naturally arise in both of them.

In data exchange, where data is shipped from a source database in order to populate a target schema, integrity constraints (ICs) imposed at the target level have to be kept satisfied. Instead of restoring the consistency of data at the target *a posteriori*, after the population process, a more appealing alternative takes into account the ICs at the target when the data mappings between the source and the target are being established and/or used. In virtual data integration there is no centralized consistency maintenance mechanism that makes the data in the mediated system satisfy certain global ICs. Again, these ICs have to be captured by the mappings between the sources and the global schema or at query time [5], when global queries are being answered. In the two scenarios, the plans for data transfer or query answering have to deal with potential inconsistencies of data.

Research around the management, mapping, and integration of database schemata is also relevant to both data exchange and integration. It is common that research in these two latter areas starts from given source and target schemas on one side, or source and global schemas on the other. Given those schemas, the problem is to design exchange or query plans. However, there is not much research that addresses the impact of schema design on the latter tasks. This perception was an additional motivation for gathering people from those different areas.

Industrial presentations gave overviews of trends in the area of metadata management for information integration in the data warehousing industry, and of the IBM® DB2® Information Integrator (now called WebSphere® Information Integrator). A hands on demo of aspects of federation, replication and enterprise search features of WebSphere Information Integrator, can be found at http://db2ii2.dfw.ibm.com/wps/myportal/!ut/p/.scr/LoggedIn.[1]

Break-out sessions were run in parallel around each of the three areas mentioned above. Acting as group discussion leaders Alex Thomo (data integration), Phokion Kolaitis (schema-mappings and data exchange), and Jan Chomicki (consistency). The following sections describe relevant research problems and trends that were identified by the working groups.

3 Data Integration

In this session, the main focus was on view-based data integration. The goal of data integration is to provide a uniform interface for querying a collection of disparate and heterogeneous data sources. The two main approaches, namely the *global-as-view* (GAV) and the *local-as-view* (LAV) were overviewed. In both, the user poses queries

[1] DB2, IBM, and WebSphere are registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

on a mediated global schema. The main difference lies in the way the data sources are represented. In GAV, each data source exposes a view defined over the local schema, and the view name is an element of the global schema. On the other hand, in LAV, the data sources expose views defined over the global schema. The pros and cons of each approach are as follows. In GAV the query answering is simple: It amounts to view unfolding. However, adding sources to the data integration system is non-trivial. In particular, given a new source, we need to figure out all the ways in which it can be used to obtain tuples for each of the views in the global schema, and this limits the ability of the GAV approach to scale to a large collection of sources. In contrast, in the LAV approach each source is described in isolation. It is the system's task to figure out (at query time) how the sources interact and how their data can be combined. In this case, query answering and query reformulation are harder, and sometimes require recursive queries over the sources.

This session focused mostly on the LAV approach. It was discussed that there is sometimes a confusion about how the answer to a query should be characterized. There are two ways of doing that. The first is syntactical: The query answer is the one that can be computed by evaluating the maximal view-based query rewriting expressed in some fixed language. The second characterization is semantical: We try to find the *certain answers*, that belong to all the answers sets that can be obtained on each database that is consistent with the views (on the global schema). The source of confusion is that these two characterizations coincide in the case of conjunctive queries and views. Thus, they have been often used interchangeably in previous works, but this coincidence is not true for more complex queries and views, such as those containing recursion.

As pointed out in [24], in the context of semistructured data, a natural and quite general fragment of recursive Datalog did emerge in the mid 1990s: The class of regular queries, whose basic element is that of regular path queries. For such queries and views to compute the certain answers is computationally hard. For example, to decide whether a tuple is a certain answer is CoNP-complete with respect to the size of data [10]. On the other hand, given a view-based rewriting expressed in some fixed language (e.g., regular language, or Datalog), computing the answer is in PTIME on the size of the data. Clearly, the answer computed by evaluating the maximal (w.r.t. to some fixed language) view-based rewriting is only a subset of the certain answers, but this is often an acceptable approximation. In the session it was concluded that the best sources of information regarding the rewriting-based answers vs. certain answers, are the seminal papers [9, 10]. Moreover, a recent important achievement was pointed out: In [11] the notion of view-based containment and equivalence for regular path queries was investigated and positively solved. By using the solution in [11], one can test, in compile time, whether a computed view-based rewriting will generate the full certain answers when evaluated on the views.

The session was concluded outlining some new promising directions for further research, among others, the optimization of view-based rewritings and data-integration in P2P systems.

4 Data Exchange and Metadata

This session focused on research issues and directions in data exchange and metadata management. A common thread in both these areas is the systematic use of *schema-mappings*, which are high-level specifications in some logical formalism that describe the relationships between schemas.

Issues in data exchange. Data exchange is the problem of taking data structured under a source schema and translating them into data structured under a target schema. Although there are clear similarities with data integration, the main difference between the two frameworks is that, given a source instance, the goal in data exchange is to actually materialize a target instance such that (i) it satisfies the specifications of the schema-mapping between the source schema and the target; (ii) it reflects the given source data as accurately as possible. The challenges in data exchange arise because typically there are more than one target instances (called *solutions*) that satisfy the specifications of the schema-mapping. This state of affairs raises both semantical issues and algorithmic issues in data exchange: Given a source instance, which solutions are better than others? Which solution should one choose to materialize? How difficult is to compute such a good solution? What is the semantics of target queries and how difficult is it to evaluate such queries?

In recent years, the study of the theoretical underpinnings of data exchange has mainly centered on data exchange settings between relational schemas and with schema-mappings specified by source-to-target tuple generating dependencies that can be thought of as global-and-local-as-view (GLAV) constraints [15, 16]. The next step is to attempt to extend this study to richer data exchange settings. One concrete direction is to study the semantics of data exchange between semistructured schemas and XML schemas, and to relate this investigation to the actual practice of existing data exchange tools, such as the CLIO system [21].

During the break-out session, there was a vigorous discussion about rethinking the semantics of query answering in data exchange. Thus far, *the certain answers* semantics has been used as the standard semantics in both data exchange and data integration, and much of the research has focused on the complexity of computing the certain answers to target queries. The definition of the certain answers is based on the entire space of solutions to the data

exchange problem. What is so sacred about this semantics? Are there meaningful alternatives to the certain answers semantics that also take into account the "pragmatics" of the situation at hand? Is there a way to gauge the "quality" of the answer to a query, in addition to mainly focusing on the complexity of computing this answer?

Issues in metadata management. Schema-mappings *are* metadata. Bernstein [4] has made a compelling case for the importance of developing both the theory and the practice of metadata management, which in this framework, is achieved by combining certain basic generic operators on schema-mappings, such as *composition*, *merge*, *match*, and *change*. Repeated combinations of these operators can produce complex transformations on schema-mappings and can be used to analyze schema evolution.

The first main challenge in metadata management is to develop rigorous semantics for each of the basic operators. Once this is achieved, the next challenge is to investigate the properties of these operators for different schema-mapping languages. One concrete issue has to do with the *closure* properties of the schema-mapping languages. For instance, is a given schema-mapping language closed under composition? In other words, can the composition of two schema-mappings be expressed in the same language used to express each of the two schema-mappings? Another issue has to do with the algorithmic properties of the basic operators and the schema-mapping languages used to express them. In particular, for which schema-mapping languages can the outputs of these operators be efficiently computed? Progress has been made in the study of several operators, including composition [20, 17] and merge [22]. Nonetheless, much more remains to be done for the remaining operators.

Another major challenge in this area is the development of better user interfaces for visualizing and manipulating schema-mappings. Techniques from other areas, such as graph drawing, may have a role to play in designing more effective user interfaces that will make it possible to achieve large-scale metadata management.

Finally, there is a need to create a suite of benchmarks to be used to carry out experiments to compare tools for data exchange and metadata management. Research in this area will undoubtedly benefit from the existence of a public-domain repository with data, schemas that have evolved over time, complex schema-mappings, and challenging queries. Building such a repository will be a real service to the community and will advance the field.

5 Handling Inconsistency of Data

The notion of *inconsistency* has been extensively studied in many contexts. In classical logic, an inconsistent set of formulas implies every formula (*triviality*). In databases, a database instance is inconsistent if it does not satisfy integrity constraints (ICs) (*constraint violation*). Those two kinds of inconsistency are closely related. Triviality is not a problem in the database context because the semantics of query answers does not take into account ICs.

Inconsistent databases arise in data integration, during long-running database activities, and in other situations in which ICs cannot or would not be enforced. In order to deal with inconsistency in a flexible manner, database research has developed different approaches that we will illustrate using a simple example.

Consider a database schema consisting of two unary relations P and Q and the IC: $\forall x.\neg(P(x) \wedge Q(x))$. Assume a database instance consists of the following facts: $\{P(a), Q(a), P(b)\}$. Under *prevention* (usual constraint enforcement), such an instance could not arise: only one of $P(a)$ and $Q(a)$ could be inserted into the database. Under *ignorance* (constraint non-enforcement), no distinction is made between $P(a)$ and $P(b)$, despite that the latter, not being involved in a constraint violation, appears to represent more reliable information. Under *isolation* [7], both $P(a)$ and $Q(a)$ would be dropped (or ignored in query answering). Under *weakening* [3, 19], $P(a)$ and $Q(a)$ would be replaced by $P(a) \vee Q(a)$ or some other form of disjunctive information. Allowing *exceptions* [6], means that the constraint is weakened to $\forall x.\neg(P(x) \wedge Q(x) \wedge x \neq a)$. *Materialized repairing* [14] produces a *repair*: a consistent instance minimally different from the original one, in this case $\{P(a), P(b)\}$ or $\{Q(a), P(b)\}$. *Virtual repairing* [1] does not change the database but rather returns query answers true in all repairs (*consistent* query answers). So the query asking for all such x that $P(x)$ is true, returns only $x = b$. Finally, under the attack/support approach [23], $P(a)$ attacks $Q(a)$ and vice versa, and thus the support for both is lower than for $P(b)$.

Research has focused on materialized or virtual repairing, and different notions of repair have been proposed, to capture the notion of minimal change in different ways [1, 8, 25]. Also, different evaluation mechanisms for computing consistent query answers have been developed and the complexity of this problem studied [2, 13, 18, 8, 12].

The following research issues in the area of inconsistent databases were viewed as important by the participants:

1. How to *generalize/integrate/parameterize* existing approaches to the computation of consistent query answers.
2. What kind of *preferences* are useful and do they help? Although preferences may reduce the number of repairs, they introduce additional minimization criteria, which may negatively influence computational complexity.
3. Should *null* and *default* values be considered in constructing repairs?
4. How to identify contexts suitable for a specific notion or repair. For example, if the database is assumed to be complete, all repairs are obtained by deleting facts. However, in the absence of such an assumption, insertions of facts should also be considered.
5. How to integrate inconsistency resolution with various

data cleaning tasks: Data normalization/standardization, record linkage and merging.

6. How to design more powerful query languages by capturing the notion of *possible* query answer (answer true in some repair) and embedding the computation of both consistent and possible answers in a first-order query language.

7. How to deal with *intractability*. Trade-offs between expressive power and complexity should be further identified and approximate answers, possibly with confidence factors, studied.

8. How to test different algorithms. Measures of inconsistency should be defined and the issue of benchmarking and systematically generating inconsistent data explored.

9. How to generalize current approaches to repairing to XML databases and various data integration scenarios (e.g., data exchange, peer-to-peer). In such scenarios an inconsistent database is not necessarily stored but virtual, which creates additional challenges for repairing and computation of consistent query answers. Some current approaches to data integration presently ignore data inconsistencies and have to be extended to deal with them.

References

1. Arenas, M., Bertossi, L. and Chomicki, J. Consistent Query Answers in Inconsistent Databases. In *ACM Symposium on Principles of Database Systems (PODS)*, pp. 68–79, 1999.
2. Arenas, M., Bertossi, L. and Chomicki, J. Answer Sets for Consistent Query Answering in Inconsistent Databases. *Theory and Practice of Logic Programming*, 3(4–5):393–424, 2003.
3. Baral, C., Kraus, S., Minker, J. and Subrahmanian, V.S. Combining Knowledge Bases Consisting of First-Order Theories. *Computational Intelligence*, 8:45–71, 1992.
4. Bernstein, P. A. Applying Model Management to Classical Meta-Data Problems. In *Conference on Innovative Data Systems Research (CIDR)*. 209–220, 2003.
5. Bertossi, L. and Bravo, L. Consistent Query Answers in Virtual Data Integration Systems. In *Inconsistency Tolerance*, Springer LNCS 3300, 2004, pp. 42-83.
6. Borgida, A. Language Features for Flexible Handling of Exceptions in Information Systems. *Proc. ACM Transactions on Database Systems*, 10(4):565–603, 1985.
7. Bry, F. Query Answering in Information Systems with Integrity Constraints. In *IFIP WG 11.5 Working Conference on Integrity and Control in Information Systems*, pp. 113–130. Chapman &Hall, 1997.
8. Calì, A., Lembo, D. and Rosati, R. On the Decidability and Complexity of Query Answering over Inconsistent and Incomplete Databases. *Proc. ACM Symposium on Principles of Database Systems (PODS)*, pp. 260–271, 2003.
9. Calvanese D., De Giacomo, G., Lenzerini, M. and Vardi, M.Y. Rewriting of Regular Expressions and Regular Path Queries. *Proc. ACM Symposium on Principles of Database Systems (PODS)*, pp. 194–204, 1999.
10. Calvanese D., De Giacomo, G., Lenzerini, M. and Vardi, M.Y. Answering Regular Path Queries Using Views. *Proc. International Conference on Data Engineering (ICDE)*, pp. 389–398, 2000.
11. Calvanese D., De Giacomo, G., Lenzerini, M. and Vardi, M.Y. View-based Query Containment. *Proc. ACM Symposium on Principles of Database Systems (PODS)*, pp. 56–67, 2003.
12. Chomicki, J. and Marcinkowski, J. Minimal-Change Integrity Maintenance Using Tuple Deletions. *Information and Computation*, 197(1-2):90-121, 2005.
13. Eiter, T., Fink, M., Greco, G. and Lembo, D. Efficient Evaluation of Logic Programs for Querying Data Integration Systems. *Proc. International Conference on Logic Programming (ICLP)*, pp. 163–177. Springer-Verlag, LNCS 2916, 2003.
14. Embury, S.M., Brandt, S.M., Robinson, J.S., Sutherland, I., Bisby, F.A., Gray, W.A., Jones, A.C. and White, R.J. Adapting integrity enforcement techniques for data reconciliation. *Information Systems*, 26(8):657–689, 2001.
15. Fagin, R., Kolaitis, Ph. G., Miller, R. J., and Popa, L. Data Exchange: Semantics and Query Answering. In *International Conference on Database Theory (ICDT)*. 207–224, 2003.
16. Fagin, R., Kolaitis, Ph. G., and Popa, L. Data Exchange: Getting to the Core. In *ACM Symposium on Principles of Database Systems (PODS)*. 90–101, 2003.
17. Fagin, R., Kolaitis, Ph.G., Popa, L., and Tan, W.-C. Composing Schema Mappings: Second-Order Dependencies to the Rescue. In *ACM Symposium on Principles of Database Systems (PODS)*. 83-94, 2004.
18. Greco, G., Greco, S. and Zumpano, E. A Logical Framework for Querying and Repairing Inconsistent Databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1389–1408, 2003.
19. Lin, J. and Mendelzon, A.O. Merging Databases under Constraints. *International Journal of Cooperative Information Systems*, 7(1):55–76, 1996.
20. Madhavan, J. and Halevy, A. Y. Composing Mappings Among Data Sources. In *International Conference on Very Large Data Bases (VLDB)*. 572–583, 2003.
21. Popa, L., Velegrakis, Y., Miller, R. J., Hernandez, M. A., and Fagin, R. Translating Web Data. In *International Conference on Very Large Data Bases (VLDB)*. 598–609, 2002.
22. Pottinger, R., and Bernstein, P. A., Merging Models Based on Given Correspondences In *International Conference on Very Large Data Bases (VLDB)*. 826-873, 2003.
23. Pradhan, S. Argumentation Databases. *Proc. International Conference on Logic Programming (ICLP)*, pp. 178–193. Springer-Verlag, LNCS 2916, 2003.
24. Vardi. M. Y. A Call to Regularity. *Proc. PCK50 - Principles of Computing & Knowledge, Paris C. Kanellakis Memorial Workshop '03*, pp. 11.
25. Wijsen, J. Condensed Representation of Database Repairs for Consistent Query Answering. *Proc. International Conference on Database Theory (ICDT)*, pages 378–393. Springer-Verlag, LNCS 2572, 2003.

Query Answering Exploiting Structural Properties*

Francesco Scarcello
DEIS, Università della Calabria, Italy
scarcello@unical.it

ABSTRACT

We review the notion of hypertree width, a measure of the degree of cyclicity of hypergraphs that is useful for identifying and solving efficiently easy instances of hard problems, by exploiting their structural properties. Indeed, a number of relevant problems from different areas, such as database theory, artificial intelligence, and game theory, are tractable when their underlying hypergraphs have small (i.e., bounded by some fixed constant) hypertree width. In particular, we describe how this notion may be used for identifying tractable classes of database queries and answering such queries in an efficient way.

1. INTRODUCTION

In this paper we deal with the fundamental problem of evaluating queries in relational databases, focusing on recently proposed techniques based on structural properties of the queries. For the sake of simplicity, we consider conjunctive queries (CQs), though most results may be easily extended to more general queries. The class CQ, equivalent in expressive power to the class of Select-Project-Join queries, is probably the most thoroughly analyzed class of database queries. Note that the great interest in conjunctive queries is also due to the fact that CQ evaluation is essentially the same problem as *conjunctive query containment* [6], which is of central importance in *view-based query processing* [2], and *constraint satisfaction*, which is one of the major problems studied in the field of AI (see, e.g., Vardi's survey paper [47] on the interactions between the areas of query evaluation and constraint satisfaction).

Recall that database management systems (DBMSs) have specialized modules, called query optimizers, looking for good ways to deal with any given query. For all commercial DBMSs, such a way is always based on *quantitative methods*: they examine a number of alternative plans for answering a query and then choose the best one, according to some cost model. These planners exploit information on the data, e.g., sizes of relations, indices, and so on. In fact, all of them compute just approximations of optimal query plans, as the optimization problem is NP-hard, in general. See [39] for a short survey of quantitative methods and for further references.

A completely different approach to query answering is based on structural properties of queries, rather than on quantitative information about data values. Exploiting such properties is possible to answer large classes of queries efficiently, that is, with a polynomial-time upper bound. The structure of a query Q is best represented by its *query hypergraph* $\mathcal{H}(Q) = (V, H)$, whose set V of vertices consists of all variables occurring in Q, and where the set H of hyperedges contains, for each query atom A, the set $var(A)$ of all variables occurring in A. As an example, consider the following query

Q_0: $ans \leftarrow s_1(A,B,D) \wedge s_2(B,C,D) \wedge s_3(B,E) \wedge s_4(D,G) \wedge s_5(E,F,G) \wedge s_6(E,H) \wedge s_7(F,I) \wedge s_8(G,J)$.

Figure 1 shows its associated hypergraph $\mathcal{H}(Q_0)$.

One of the most important and deeply studied class of tractable queries is the class of *acyclic queries* [5, 7, 9, 14, 28, 33, 37, 48, 49]. It was shown that acyclic queries coincide with the *tree queries* [4], see also [1, 30, 43]. The latter are queries whose query hypergraph has a *join tree* (or *join forest*) (see Section 3 for a formal definition). By well-known results of Yannakakis [48], acyclic conjunctive queries are efficiently solvable. More precisely, all answers of an acyclic conjunctive query can be computed in time polynomial in the combined size of the input and the output. This is the best possible result, because in general the answer of a query may contain an exponential number of tuples. Recall that, for cyclic queries, even computing small outputs, e.g. just one tuple, or checking whether the answer of a query is non-empty (Boolean queries) requires exponential time (unless P = NP) [6].

Therefore, many attempts have been made in the literature for extending the good results about acyclic conjunctive queries to relevant classes of *nearly acyclic* queries. We call these techniques *structural query decomposition methods*,[1] because they are based on the acyclicization of cyclic (hyper)graphs. More precisely, each method specifies how appropriately transforming a conjunctive query into an equivalent tree query (i.e., acyclic query given in form of a join tree), by organizing its atoms into a polynomial number of clusters, and suitably arranging the clusters as a tree (see Figure 1). Each cluster contains a number of atoms. After performing the join of the relations corresponding to the atoms jointly contained in each cluster, we obtain a join tree of an acyclic query which is equivalent to the original query. The resulting query can be answered in output-polynomial time by Yannakakis's algorithm. Thus, in case of a Boolean query, it can be answered in polynomial time. The tree of atom-clusters produced by a structural query decomposition method on a given query Q is referred to as the *decomposition* of Q. Figure 1 also shows two possible decompositions of our example query Q_0. A decomposition of Q can be seen as a query plan for Q, requiring to first evaluate the join of each cluster, and then to process the resulting join tree in a bottom-up fashion (following Yannakakis's algorithm).

*__Database Principles Column.__ Column editor: Leonid Libkin, Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3H5, Canada. E-mail:libkin@cs.toronto.edu

[1] In the field of constraint satisfaction, the same notion is known as *structural CSP decomposition method*, cf. [15].

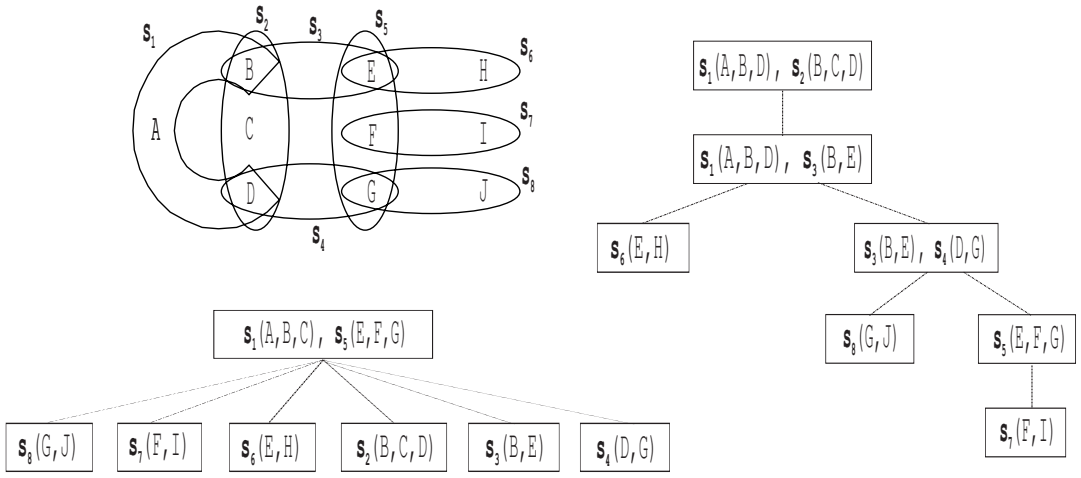


Figure 1: Hypergraph $\mathcal{H}(Q_0)$ (left), two hypertree decompositions of width 2 of $\mathcal{H}(Q_0)$ (right and bottom).

Thus, the efficiency of a structural decomposition method essentially depends on the maximum size of the produced clusters, measured (according to the chosen decomposition method) either in terms of the number of variables or in terms of the number of atoms. For a given decomposition, this size is referred-to as the *width* of the decomposition. For example, if we adopt the number of atoms, then the width of both decompositions shown in Figure 1 is 2. Intuitively, the complexity of transforming a given decomposition into an equivalent tree query is exponential in its width w. In fact, the evaluation cost of each of the (polynomially many) clusters is bounded by the cost of performing the (at most) w joins of its relations, which is in turn bounded by $O(|r_{max}|^{w-1} \log |r_{max}|)$, where $|r_{max}|$ denotes the size of the largest relation r_{max} in the database. The overall cost (transformation+evaluation of the resulting acyclic query) is thus $O(v|r_{max}|^{w-1} \log |r_{max}|)$, where v is the number of vertices of the decomposition tree. It is worthwhile noting that, for queries involving many atoms, exploiting such a structural information may lead to a quite remarkable computational saving. For instance, the above upper bound is $O(7|r_{max}| \log |r_{max}|)$ for the query in Figure 1, whereas typical query answering algorithms would take $O(|r_{max}|^7 \log |r_{max}|)$ time, in the worst case.

In general, a rough upper bound for the cost of answering a given query Q according to any structural method D is given by $O(n^{w+1} \log n)$, where w is the D-width of Q and n is the total size of the input problem, that is, the size of the query and of the database encoding [15]. Therefore, once we fix a bound k for such a width, the structural method D identifies a class of queries that can be answered in polynomial time, namely, the class of all queries having k-bounded D-width (i.e., D-width at most k).[2] The main structural decomposition methods are based on the notions of Biconnected Components [11], Tree Decompositions [35, 7, 28, 8, 25, 10], Hinge Decompositions [26], and Hypertree Decompositions [18, 19, 21, 40].

Among them, the Hypertree Decomposition Method (HYPERTREE) seems to be the most powerful method, as a large class of cyclic queries has a low hypertree-width, and in fact it strongly generalizes all other structural methods [15]. More precisely, this means that every class of queries that is recognized as tractable according to any structural method D (has k-bounded D-width), is also tractable according to HYPERTREE (has k-bounded HYPERTREE-width), and that there are classes of queries that are tractable according to HYPERTREE, but not tractable w.r.t. D (have unbounded D-width). Moreover, for any fixed $k > 0$, deciding whether a hypergraph has hypertree width at most k is feasible in polynomial time, and is actually highly parallelizable, as this problem belongs to LOGCFL [19] (See [36, 17], for properties and characterizations of this complexity class). In fact, it has been conjectured that a class of queries is tractable if and only the cores of their structures have bounded hypertree width (under some widely believed complexity-theoretic assumptions) [24]. The first part of this paper is devoted to the presentation of the main results about hypertree decompositions.

Despite their very nice computational properties, all the above structural decomposition methods, including Hypertree Decomposition, are often unsuited for some real-world applications. For instance, in a practical context, one may prefer query plans (i.e., minimum-width decompositions) which minimize the number of clusters having the largest cardinality. Even more importantly, decomposition methods focus "only" on structural features, while they completely disregard "quantitative" aspects of the query, that may dramatically affect the query-evaluation time. For instance, while answering a query, the computation of an *arbitrary* hypertree decomposition (having minimum width) could not be satisfactory, since it does not take into account important quantitative factors, such as relation sizes, attribute selectivity, and so on. These factors are flattened in the query hypergraph (which considers only the query structure), while their suitable exploitation can significantly reduce the cost of query evaluation.

On the other hand, query optimizers of commercial DBMSs are based solely on quantitative methods and do not care of structural properties at all. Indeed, all the commercial DBMSs restrict the search space of query plans to very simple structures (e.g., left-deep trees), and then try to find the best plans among them, by estimating their evaluation costs, exploiting quantitative information on the input database. It follows that, on some low-width queries with a guaranteed polynomial-time evaluation upper-bound, they may also take time $O(n^\ell)$, which is exponential in the length ℓ of

[2] Intuitively, the D-width of a query Q is the minimum width of the decompositions of Q obtainable by method D.

the query, rather than on its width. On some relevant applications with many atoms involved, this may lead to unacceptable costs. For instance, consider the problem of the population and refreshing of cubes in data warehouse initialization and management. Periodically, a number of batch queries are executed on the reconciled operational database. Note that these queries are typically very different from OLAP queries. Indeed, while the latter queries are executed on star schemes (or similar simple schemes), these populating queries usually span several tables in the reconciled scheme in order to update both dimension and fact tables. Thus, they are very often long queries involving many join operations, plus selections, projections and, possibly, grouping and aggregate operators. In this context, the choice of a good query-execution strategy is therefore particularly relevant, because the differences among execution times can be several orders of magnitude large. In fact, very often such queries are not very intricate and have low hypertree width, though they are not necessarily acyclic.

To overcome the above mentioned drawbacks of both approaches, we proposed an extension of hypertree decompositions, in order to combine this structural decomposition method with quantitative approaches [41]. In the second part of this paper, we review the main results on this generalized notion of HYPERTREE, where hypertree decompositions are equipped with polynomial-time weight functions that may encode quantitative aspects of the query database, or other additional requirements. In general, computing a minimal weighted-decomposition is harder than computing a standard decomposition. However, we present a class of functions, called *tree aggregation functions* (TAFs), which is useful for query optimization and easy to deal with.

We describe how the notion of weighted hypertree decomposition can be used for generating effective query plans for the evaluation of conjunctive queries, by combining structural and quantitative information. We also briefly report some results of an ongoing experimental activity, showing that this hybrid approach may in fact lead to significant computational savings.

2. QUERIES AND ACYCLIC HYPERGRAPHS

We will adopt the standard convention [1, 43] of identifying a relational database instance with a logical theory consisting of ground facts. Thus, a tuple $\langle a_1, \ldots a_k \rangle$, belonging to relation r, will be identified with the ground atom $r(a_1, \ldots, a_k)$. The fact that a tuple $\langle a_1, \ldots, a_k \rangle$ belongs to relation r of a database instance **DB** is thus simply denoted by $r(a_1, \ldots, a_k) \in \mathbf{DB}$.

A (rule-based) *conjunctive query* Q on a database schema $DS = \{R_1, \ldots, R_m\}$ consists of a rule of the form

$$Q: \quad ans(\mathbf{u}) \leftarrow r_1(\mathbf{u_1}) \wedge \cdots \wedge r_n(\mathbf{u_n}),$$

where $n \geq 0$; $r_1, \ldots r_n$ are relation names (not necessarily distinct) of DS; ans is a relation name not in DS; and $\mathbf{u}, \mathbf{u_1}, \ldots, \mathbf{u_n}$ are lists of terms (i.e., variables or constants) of appropriate length. The set of variables occurring in Q is denoted by $var(Q)$. The set of atoms contained in the body of Q is referred to as $atoms(Q)$.

The *answer* of Q on a database instance **DB** with associated universe U, consists of a relation ans, whose arity is equal to the length of $\mathbf{u}$, defined as follows. Relation ans contains all tuples $\mathbf{u}\theta$ such that $\theta : var(Q) \longrightarrow U$ is a substitution replacing each variable in $var(Q)$ by a value of U and such that for $1 \leq i \leq n$, $r_i(\mathbf{u_i})\theta \in \mathbf{DB}$. (For an atom A, $A\theta$ denotes the atom obtained from A by uniformly substituting $\theta(X)$ for each variable X occurring in A.)

If Q is a conjunctive query, we define the hypergraph $H(Q) = (V, E)$ associated to Q as follows. The set of vertices V, denoted by $var(H(Q))$, consists of all variables occurring in Q. The set E, denoted by $edges(H(Q))$, contains for each atom $r_i(\mathbf{u_i})$ in the body of Q a hyperedge consisting of all variables occurring in $\mathbf{u_i}$. Note that the cardinality of $edges(H(Q))$ can be smaller than the cardinality of *atoms(Q)*, because two query atoms having exactly the same set of variables in their arguments give rise to only one edge in $edges(H(Q))$. For example, the three query atoms $r(X, Y)$, $r(Y, X)$, and $s(X, X, Y)$ all correspond to a unique hyperedge $\{X, Y\}$.

A query Q is acyclic if and only if its hypergraph $H(Q)$ is acyclic or, equivalently, if it has has a join forest. A *join forest* for the hypergraph $H(Q)$ is a forest G whose set of vertices V_G is the set $edges(H(Q))$ and such that, for each pair of hyperedges h_1 and h_2 in V_G having variables in common (i.e., such that $h_1 \cap h_2 \neq \emptyset$), the following conditions hold:

1. h_1 and h_2 belong to the same connected component of G, and
2. all variables common to h_1 and h_2 occur in every vertex on the (unique) path in G from h_1 to h_2.

If G is a tree, then it is called a *join tree* for $H(Q)$.

Intuitively, the efficient behavior of acyclic instances is due to the fact that they can be evaluated by processing any of their join trees bottom-up by performing upward semijoins, thus keeping the size of the intermediate relations small (while it could become exponential, if regular join were performed).

Let us recall the highly desirable computational properties of acyclic queries:

1. Acyclic instances can be efficiently solved. Yannakakis provided a (sequential) polynomial time algorithm for Boolean acyclic queries[3]. Moreover, he showed that the answer of a non-Boolean acyclic conjunctive query can be *computed* in time polynomial in the combined size of the input instance and of the output relation [48].

2. We have shown that answering queries is highly parallelizable on acyclic queries, as this problem (actually, the decision problem of answering Boolean queries) is complete for the low complexity class LOGCFL [18]. Efficient parallel algorithms for Boolean and non-Boolean queries have been proposed in [18] and [16]. They run on parallel database machines that exploit the *inter-operation parallelism* [44], i.e., machines that execute different relational operations in parallel. These algorithms can be also employed for solving acyclic queries efficiently in a distributed environment.

3. Acyclicity is efficiently recognizable: deciding whether a hypergraph is acyclic is feasible in linear time [42] and belongs to the class L (deterministic logspace). The latter result is

[3]Note that, since both the database **DB** and the query Q are part of an input-instance, what we are considering is the *combined complexity* of the query [46].

new: it follows from the fact that hypergraph acyclicity belongs to SL [17], and from the very recent proof that SL is in fact equal to L [34].

3. HYPERTREE DECOMPOSITIONS

We recall the formal definition and the most important results about *hypertree width* and *hypertree decompositions*.

A *hypertree for a hypergraph* $\mathcal{H}$ is a triple $\langle T, \chi, \lambda \rangle$, where $T = (N, E)$ is a rooted tree, and χ and λ are labeling functions which associate to each vertex $p \in N$ two sets $\chi(p) \subseteq var(\mathcal{H})$ and $\lambda(p) \subseteq edges(\mathcal{H})$. The *width* of a hypertree is the cardinality of its largest λ label, i.e., $max_{p \in N}|\lambda(p)|$.

We denote the set of vertices of any rooted tree T by $vertices(T)$, and its root by $root(T)$. Moreover, for any $p \in vertices(T)$, T_p denotes the subtree of T rooted at p. If T' is a subtree of T, we define $\chi(T') = \bigcup_{v \in vertices(T')} \chi(v)$.

Definition 3.1 *[21]*A *generalized hypertree decomposition* of a hypergraph $\mathcal{H}$ is a hypertree $HD = \langle T, \chi, \lambda \rangle$ for $\mathcal{H}$ which satisfies the following conditions:

1. *For each edge $h \in edges(\mathcal{H})$, all of its variables occur together in some vertex of the decomposition tree*, that is, there exists $p \in vertices(T)$ such that $h \subseteq \chi(p)$ (we say that p *covers* h).
2. *Connectedness Condition*: for each variable $Y \in var(\mathcal{H})$, the set $\{p \in vertices(T) \mid Y \in \chi(p)\}$ induces a (connected) subtree of T.
3. For each vertex $p \in vertices(T)$, *variables in the χ labeling should belong to edges in the λ labeling*, that is, $\chi(p) \subseteq var(\lambda(p))$.

A *hypertree decomposition* is a generalized hypertree decomposition that satisfies the following additional condition:

4. *Special Descendant Condition*: for each $p \in vertices(T)$, $var(\lambda(p)) \cap \chi(T_p) \subseteq \chi(p)$.

The HYPERTREE *width* $hw(\mathcal{H})$ (resp., generalized hypertree width $ghw(\mathcal{H})$) of $\mathcal{H}$ is the minimum width over all its hypertree decompositions (resp., generalized hypertree decompositions).

An edge $h \in edges(\mathcal{H})$ is *strongly covered* in *HD* if there exists $p \in vertices(T)$ such that $var(h) \subseteq \chi(p)$ and $h \in \lambda(p)$. In this case, we say that p strongly covers h. A decomposition *HD* of hypergraph $\mathcal{H}$ is a *complete decomposition* of $\mathcal{H}$ if every edge of $\mathcal{H}$ is strongly covered in *HD*. From any (generalized) hypertree decomposition *HD* of $\mathcal{H}$, we can easily compute a complete (generalized) hypertree decomposition of $\mathcal{H}$ having the same width.

Note that the notions of hypertree width and generalized hypertree width are true generalizations of acyclicity, as the acyclic hypergraphs are precisely those hypergraphs having hypertree width and generalized hypertree width one. In particular, as we will see in the next section, the classes of conjunctive queries having bounded (generalized) hypertree width have the same desirable computational properties as acyclic queries [19].

At first glance, a generalized hypertree decomposition of a hypergraph may simply be viewed as a clustering of the hyperedges (i.e., query atoms) where the classical connectedness condition of join trees holds. However, a generalized hypertree decomposition may deviate in two ways from this principle: **(1)** A hyperedge already used in some cluster may be reused in some other cluster; **(2)** Some variables occurring in reused hyperedges are not required to fulfill any condition.

For a better understanding of this notion, let us focus on the two labels associated with each vertex p: the set of hyperedges $\lambda(p)$, and the set of *effective* variables $\chi(p)$, which are subject to the connectedness condition (2). Note that all variables that appear in the hyperedges of $\lambda(p)$ but that are not included in $\chi(p)$ are "ineffective" for v and do not count w.r.t. the connectedness condition. Thus, the χ labeling plays the crucial role of providing a join-tree like re-arranging of all connections among variables. Besides the connectedness condition, this re-arranging should fulfill the fundamental Condition 1: every hyperedge (i.e., query atom, in our context) has to be properly considered in the decomposition, as for graph edges in tree-decompositions and for hyperedges in join trees (where this condition is actually even stronger, as hyperedges are in a one-to-one correspondence with vertices of the tree). Since the only relevant variables are those contained in the χ labels of vertices in the decomposition tree, the λ labels are "just" in charge of covering such relevant variables (Condition 3) with as few hyperedges as possible. Indeed, the width of the decomposition is determined by the largest λ label in the tree. This is the most important novelty of this approach, and comes from the specific properties of hypergraph-based problems, where hyperedges often play a predominant role. For instance, think of our database framework: the cost of evaluating a natural join operation with k atoms (read: k hyperedges) is $O(n^{k-1} \log n)$, no matter of the number of variables occurring in the query.

Example 3.2 Consider the following conjunctive query Q_1:

$$\begin{aligned} ans \;\leftarrow\; & a(S, X, X', C, F) \wedge b(S, Y, Y', C', F') \\ & \wedge c(C, C', Z) \wedge d(X, Z) \wedge \\ & e(Y, Z) \wedge f(F, F', Z') \wedge g(X', Z') \wedge \\ & h(Y', Z') \wedge j(J, X, Y, X', Y'). \end{aligned}$$

Let $\mathcal{H}_1$ be the hypergraph associated to Q_1. Since $\mathcal{H}_1$ is cyclic, $hw(\mathcal{H}_1) > 1$ holds. Figure 2 shows a (complete) hypertree decomposition HD_1 of $\mathcal{H}_1$ having width 2, hence $hw(\mathcal{H}_1) = 2$.

In order to help the intuition, Figure 3 shows an alternative representation of this decomposition, called *atom* (or *hyperedge*) *representation* [19]: each node p in the tree is labeled by a set of atoms representing $\lambda(p)$; $\chi(p)$ is the set of all variables, distinct from '_', appearing in these hyperedges. Thus, in this representation, possible occurrences of the anonymous variable '_' take the place of variables in $var(\lambda(p)) - \chi(p)$.

Another example is depicted in Figure 1, which shows two hypertree decompositions of query Q_0 in Section 1. Both decompositions have width two and are complete decompositions of Q_0. □

Let k be a fixed positive integer. We say that a CQ instance I has k-bounded (generalized) hypertree width if $(g)hw(\mathcal{H}(I)) \leq k$. A class of queries has bounded (generalized) hypertree width if there is some $k \geq 1$ such that all instances in the class have k-bounded (generalized) hypertree width.

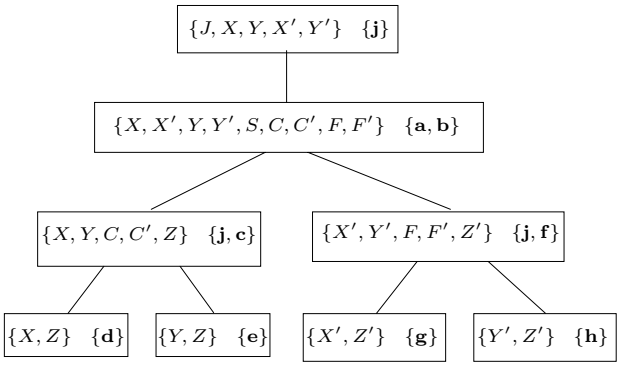


Figure 2: A 2-width hypertree decomposition of hypergraph $\mathcal{H}_1$ in Example 3.2

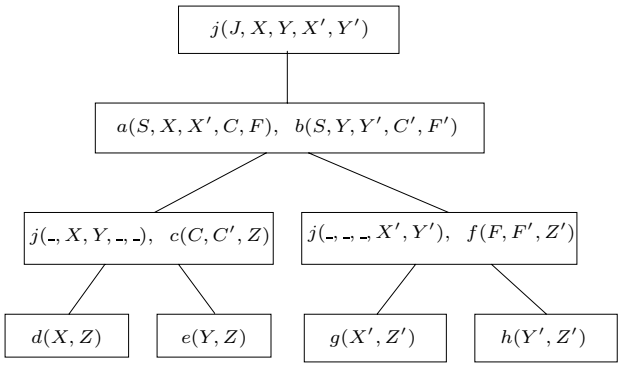


Figure 3: Atom representation of the hypertree decomposition in Figure 2

Clearly enough, choosing a tree and a clever combination of χ and λ labeling for its vertices in order to get a decomposition below a fixed threshold width k is not that easy, and is definitely more difficult than computing a simple tree decomposition, where only variables are associated with each vertex. In fact, the tractability of generalized hypertree width is an interesting open problem, as no polynomial time algorithm is known for deciding whether a hypergraph has generalized hypertree width at most k, for any fixed $k \geq 2$.

It is thus very nice and somehow surprising that dealing with the hypertree width is a very easy task. More precisely, for any fixed $k \geq 1$, deciding whether a given hypergraph has hypertree width at most k is in LOGCFL, and thus it is a tractable and highly parallelizable problem. Correspondingly, the search problem of computing a k-bounded hypertree decomposition belongs to the functional version of LOGCFL, which is $\mathrm{L}^{\mathrm{LOGCFL}}$ [19]. See the Hypertree Decomposition Homepage [40], for available implementations of algorithms for computing hypertree decompositions, and further links to heuristics and other papers on this subject.

Let us briefly discuss the only difference of hypertree decomposition with respect to generalized hypertree decomposition, that is, the *descendant condition* (Condition 4 in Definition 3.1). Consider a vertex p of a hypertree decomposition and a hyperedge $h \in \lambda(p)$ such that some variables $\bar{X} \subseteq h$ occur in the χ labeling of some vertices in the subtree T_p rooted at p. Then, according to this condition, these variables must occur in $\chi(p)$, too. This means, intuitively, that we have to deal with variables in $\bar{X}$ at this point of the decomposition tree, if we want to put h in $\lambda(p)$. For instance, as a consequence of this condition, for the root r of any hypertree decomposition we always have $\chi(r) = var(\lambda(r))$. However, once a hyperedge has been covered by some vertex of the decomposition tree, any subset of its variables can be used freely in order to decompose the remaining cycles in the hypergraph.

To shed more light on this restriction, consider what happens in the related hypergraph-based notions: in query decompositions [7], all variables are relevant; at the opposite side, in generalized hypertree decompositions, we can choose as relevant variables any subset of variables occurring in λ, without any limitation; in hypertree decompositions, we can choose any subset of relevant variables as long as the above descendant condition is satisfied. Therefore, the notion of hypertree width is clearly more powerful than the (intractable) notion of query width, but less general than the (probably intractable) notion of generalized hypertree width, which is the most liberal notion.

For instance, look at Figure 3: the variables in the hyperedge corresponding to atom j in $\mathcal{H}_1$ are jointly included only in the root of the decomposition, while we exploit two different subsets of this hyperedge in the rest of the decomposition tree. Note that the descendant condition is satisfied. Take the vertex at level 2, on the left: the variables j, X' and Y' are not in the χ label of this vertex (they are replaced by the anonymous variable '_'), but they do not occur anymore in the subtree rooted at this vertex. On the other hand, if we were forced to take all the variables occurring in every atom in the decomposition tree, it would not be possible to find a decomposition of width 2. Indeed, j is the only atom containing both pairs X, Y and X', Y', and it cannot be used again entirely, for its variable J cannot occur below the vertex labeled by a and b, otherwise it would violate the connectedness condition (i.e., Condition 2 of Definition 3.1). In fact, every query decomposition of this hypergraph has width 3, while the hypertree width is 2. In this case the generalized hypertree width is 2, as well, but in general it may be less than the hypertree width. However, after a recent interesting result by Adler et al. [3], the difference of these two notions of width is within a constant factor: for any hypergraph $\mathcal{H}$, $ghw(\mathcal{H}) \leq hw(\mathcal{H}) \leq 3ghw(\mathcal{H}) + 1$. It follows that a class of hypergraphs has bounded generalized hypertree width if and only if it has bounded hypertree width, and thus the two notions identify the same set of tractable classes.

Though the formal definition of hypertree width is rather involved, it is worthwhile noting that this notion has very natural characterizations in terms of games and logics [21]:

- **The robber and marshals game (R&Ms game).** It is played by one robber and a number of marshals on a hypergraph. The robber moves on variables, while marshals move on hyperedges. At each step, any marshal controls an entire hyperedge. During a move of the marshals from the set of hyperedges E to to the set of hyperedges E', the robber cannot pass through the vertices in $B = (\cup E) \cap (\cup E')$, where, for a set of hyperedges F, $\cup F$ denotes the union of all hyperedges in F. Intuitively, the vertices in B are those not released by the marshals during the move. As in the monotonic robber and cops game defined for treewidth [38], it is required that the marshals capture the robber by monotonically shrinking the moving space of the robber. The game is

won by the marshals if they corner the robber somewhere in the hypergraph. A hypergraph $\mathcal{H}$ has k-bounded hypertree width if and only if k marshals win the R&Ms game on $\mathcal{H}$.

- **Logical characterization of hypertree width.** Let L denote the existential conjunctive fragment of positive first order logic (FO). Then, the class of queries having k-bounded hypertree width is equivalent to the k-guarded fragment of L, denoted by $\mathrm{GF}_k(\mathrm{L})$. Roughly, we say that a formula Φ belongs to $\mathrm{GF}_k(\mathrm{L})$ if, for any subformula ϕ of Φ, there is a conjunction of up to k atoms jointly acting as a guard, that is, covering the free variables of ϕ. Note that this notion is related to the *loosely guarded fragment* as defined (in the context of full FO) by Van Benthem [45], where an arbitrary number of atoms may jointly act as guards (see also [23]).

3.1 Query Decompositions and Query Plans

In this section we describe the basic idea to exploit (generalized) hypertree decompositions for answering conjunctive queries.

Let $k \geq 1$ be a fixed constant, Q a conjunctive query over a database **DB**, and $HD = \langle T, \chi, \lambda \rangle$ a generalized hypertree decomposition of Q of width $w \leq k$. Then, we can answer Q in two steps:

1. For each vertex $p \in vertices(T)$, compute the join operations among relations occurring together in $\lambda(p)$, and project onto the variables in $\chi(p)$. At the end of this phase, the conjunction of these intermediate results forms an acyclic conjunctive query, say Q', equivalent to Q. Moreover, the decomposition tree T represents a join tree of Q'.
2. Answer Q', and hence Q, by using any algorithm for acyclic queries, e.g. Yannakakis's algorithm.

For instance, Figure 4 shows the tree $\mathcal{JT}_1$ obtained after Step 1 above, from the query Q_1 in Example 3.2 and the generalized hypertree decomposition in Figure 3. E.g. observe how the vertex labeled by atom p_3 is built. It comes from the join of atoms j and c (occurring in its corresponding vertex in Figure 3), and from the subsequent projection onto the variables X, Y, C, C', and Z (belonging to the χ label of that vertex). By construction, $\mathcal{JT}_1$ satisfies the connectedness condition. Therefore, the conjunction of atoms labeling this tree is an acyclic query, say Q'_1, such that $\mathcal{JT}_1$ is one of its join trees. Moreover, it is easy to see that Q'_1 has the same answer as Q_1 [19].

Step 1 is feasible in $O(m|r_{max}|^{w-1} \log |r_{max}|)$ time, where m is the number of vertices of T, and r_{max} is the relation of **DB** having the largest size. In fact, for Boolean queries, Yannakakis's algorithm in Step 2 does not take more time than Step 1, and thus its cost is an upper bound for the entire query evaluation process. For non-Boolean queries, Yannakakis's algorithm works in time polynomial in the combined size of the input and of the output, and thus we should add to the above cost a term that depends on the answer of the given query (which may be exponential w.r.t. the input size). For instance, if we consider query Q_1, the above upper bound is $O(7|r_{max}| \log |r_{max}|)$, whereas typical query answering algorithms (which do not exploit structural properties) would take $O(|r_{max}|^7 \log |r_{max}|)$ time, in the worst case.

It has been observed that, according to Definition 3.1, a hypergraph may have some (usually) undesirable hypertree decompositions [19], possibly with a large number m of vertices in the decomposition tree. For instance, a decomposition may contain two vertices with exactly the same labels. Therefore, a *normal form* for hypertree decompositions has been defined in [19], and then strengthened in [41], in order to avoid such kind of redundancies. Hypertree decompositions in normal form having width at most k may be computed in time polynomial in the size of the given hypergraph $\mathcal{H}$ (but exponential in the parameter k). The number m of vertices cannot exceed the number of variables in $\mathcal{H}$, and is typically much smaller. Moreover, $\mathcal{H}$ has a hypertree decomposition of width w if and only if it has a normal-form hypertree decomposition of the same width w.

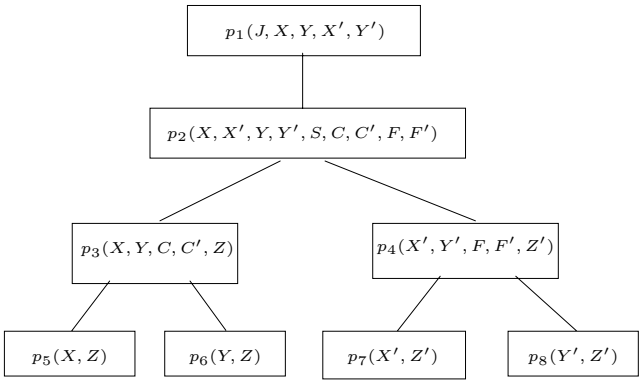


Figure 4: Join tree $\mathcal{JT}_1$ computed for query Q'_1

It follows that, for any fixed $k \geq 1$, the class of all queries having k-bounded hypertree width may be answered in polynomial time (actually, in input-output polynomial time, for non-Boolean queries). Indeed, given a query Q, both computing a hypertree decomposition HD of width at most k of $\mathcal{H}(Q)$, and then answering Q exploiting HD are polynomial-time tasks.

As far as generalized hypertree decompositions are concerned, we currently miss a polynomial-time algorithm for recognizing queries having k-bounded generalized hypertree-width. However, there is a great deal of interest in these decompositions, and some first results are coming. For instance, some very good heuristics for computing generalized hypertree decompositions are described in [29, 32].

4. WEIGHTED HYPERTREE DECOMPOSITIONS

As described in the previous section, given a query Q on a database **DB** and a small-width decomposition HD for Q, we know that there is a polynomial time upper bound for answering Q, while in general this problem is NP-hard and all the available algorithms requires exponential time, in the worst case. However, HD is not just a theoretical indication of tractability for Q. Rather, the above two steps for evaluating Q actually represent a query plan for it, though not completely specified. For instance, no actual join method (merge, nested-loop, etc.) is chosen, but this final more physical phase can be easily implemented using well-known database techniques. We remark that such optimizations are executed just on relations belonging to the same vertex, and hence on w relations at most, if w is the width of HD. Thus, also optimal methods based on dynamic programming or sophisticated heuristics can be employed, as the size of the problem is small.

The remaining interesting problem is before this evaluation phase, where we have to compute a decomposition for $\mathcal{H}(Q)$. Indeed, in general there is an exponential number of hypertree decompositions of a hypergraph. Every decomposition encodes a way of aggregating groups of atoms and arranging them in a tree-like fashion. As far as the polynomial-time upper bound is concerned, we may be happy with any minimum-width decomposition. However, in practical real-world applications we have to exploit all available information. In particular, for database queries, we cannot get rid of information on the database **DB**. Indeed, looking only at the query structure is not the best we can do, if we may additionally exploit the knowledge of relation sizes, attribute selectivity, and so on.

4.1 Minimal Decompositions

In this section, we thus consider hypertree decompositions with an associated weight, which encodes our preferences, and allows us to take into account further requirements, besides the width. We will see how to answer queries more efficiently, by looking for their best decompositions.

Formally, given a hypergraph $\mathcal{H}$, a *hypertree weighting function* (short: HWF) $\omega_{\mathcal{H}}$ is any polynomial-time function that maps each generalized hypertree decomposition $HD = \langle T, \chi, \lambda \rangle$ of $\mathcal{H}$ to a real number, called the *weight* of HD.

For instance, a very simple HWF is the function $\omega^{w}_{\mathcal{H}}(HD) = \max_{p \in vertices(T)} |\lambda(p)|$, that weights a decomposition HD just on the basis of its worse vertex, that is the vertex with the largest λ label, which also determines the width of the decomposition.

In many applications, finding such a decomposition having the minimum width is not the best we can do. We can think of minimizing the number of vertices having the largest width w and, for decompositions having the same numbers of such vertices, minimizing the number of vertices having width $w-1$, and continuing so on, in a lexicographical way. To this end, we can define the HWF $\omega^{lex}_{\mathcal{H}}(HD) = \sum_{i=1}^{w} |\{p \in N \text{ such that } |\lambda(p)| = i\}| \times B^{i-1}$, where $N = vertices(T)$, $B = |edges(\mathcal{H})| + 1$, and w is the width of HD. Note that any output of this function can be represented in a compact way as a radix B number of length w, which is clearly bounded by the number of edges in $\mathcal{H}$. Consider again the query Q_0 of the Introduction, and the hypertree decomposition, say HD', of $\mathcal{H}(Q_0)$ shown in Figure 1, on the right. It is easy to see that HD' is not the best decomposition w.r.t. $\omega^{lex}_{\mathcal{H}}$ and the class of hypertree decompositions in normal form. Indeed, $\omega^{lex}_{\mathcal{H}}(HD') = 4 \times 9^0 + 3 \times 9^1$, and thus the decomposition HD'' shown on the bottom of Figure 1 is better than HD', as $\omega^{lex}_{\mathcal{H}}(HD'') = 6 \times 9^0 + 1 \times 9^1$.

Let $k > 0$ be a fixed integer and $\mathcal{H}$ a hypergraph. We define the class $kHD_{\mathcal{H}}$ (resp., $kNFD_{\mathcal{H}}$) as the set of all hypertree decompositions (resp., normal-form hypertree decompositions) of $\mathcal{H}$ having width at most k.

Definition 4.1 *[41]*Let $\mathcal{H}$ be a hypergraph, $\omega_{\mathcal{H}}$ a weighting function, and $\mathcal{C}_{\mathcal{H}}$ a class of generalized hypertree decompositions of $\mathcal{H}$. Then, a decomposition $HD \in \mathcal{C}_{\mathcal{H}}$ is *minimal* w.r.t. $\omega_{\mathcal{H}}$ and $\mathcal{C}_{\mathcal{H}}$, denoted by $[\omega_{\mathcal{H}}, \mathcal{C}_{\mathcal{H}}]$-$minimal$, if there is no $HD' \in \mathcal{C}_{\mathcal{H}}$ such that $\omega_{\mathcal{H}}(HD') < \omega_{\mathcal{H}}(HD)$. □

For instance, the $[\omega^{w}_{\mathcal{H}}, kHD_{\mathcal{H}}]$-$minimal$ decompositions are exactly the k-bounded hypertree decompositions having the minimum possible width, while the $[\omega^{lex}_{\mathcal{H}}, kHD_{\mathcal{H}}]$-$minimal$ hypertree decompositions are a subset of them, corresponding to the lexicographically minimal decompositions described above.

It is not difficult to show that, for general weighting functions, the computation of minimal decompositions is a difficult problem even if we consider just bounded hypertree decompositions [41]. We thus restrict our attention to simpler HWFs.

Let $\langle \mathbb{R}^+, \oplus, \min, \bot, +\infty \rangle$ be a *semiring*, that is, $\oplus$ is a commutative, associative, and closed binary operator, $\bot$ is the neuter element for $\oplus$ (e.g., 0 for $+$, 1 for $\times$, etc.) and the absorbing element for $\min$, and $\min$ distributes over $\oplus$.[4] Given a function g and a set of elements $S = \{p_1, ..., p_n\}$, we denote by $\bigoplus_{p_i \in S} g(p_i)$ the value $g(p_1) \oplus \ldots \oplus g(p_n)$.

Definition 4.2 *[41]*Let $\mathcal{H}$ be a hypergraph. Then, a *tree aggregation function* (short: TAF) is any hypertree weighting function of the form

$$\mathsf{F}^{\oplus,v,e}_{\mathcal{H}}(HD) = \bigoplus_{p \in N} \big(v_{\mathcal{H}}(p) \ \oplus \bigoplus_{(p,p') \in E} e_{\mathcal{H}}(p,p') \big),$$

associating an $\mathbb{R}^+$ value to the hypertree decomposition $HD = \langle (N,E), \chi, \lambda \rangle$, where $v_{\mathcal{H}} : N \mapsto \mathbb{R}^+$ and $e_{\mathcal{H}} : N \times N \mapsto \mathbb{R}^+$ are two polynomial functions evaluating vertices and edges of hypertrees, respectively. □

We next focus on a tree aggregation function that is useful for query optimization. We refer the interested reader to [41] for further examples and applications.

Given a query Q over a database **DB**, let $HD = \langle T, \chi, \lambda \rangle$ be a hypertree decomposition in normal form for $\mathcal{H}(Q)$. For any vertex p of T, let $E(p)$ denote the relational expression $E(p) = \bowtie_{h \in \lambda(p)} \prod_{\chi(p)} rel(h)$, i.e., the join of all relations in **DB** corresponding to hyperedges in $\lambda(p)$, suitably projected onto the variables in $\chi(p)$. Given also an incoming node p' of p in the decomposition HD, we define $v^*_{\mathcal{H}(Q)}(p)$ and $e^*_{\mathcal{H}(Q)}(p,p')$ as follows:

- $v^*_{\mathcal{H}(Q)}(p)$ is the estimate of the cost of evaluating the expression $E(p)$, and
- $e^*_{\mathcal{H}(Q)}(p,p')$ is the estimate of the cost of evaluating the semi-join $E(p) \ltimes E(p')$.

Let $cost_{\mathcal{H}(Q)}$ be the TAF $\mathsf{F}^{+,v^*,e^*}_{\mathcal{H}(Q)}(HD)$, determined by the above functions. Intuitively, $cost_{\mathcal{H}(Q)}$ weights the hypertree decompositions of the query hypergraph $\mathcal{H}(Q)$ in such a way that minimal hypertree decompositions correspond to "optimal" query evaluation plans for Q over **DB**. Note that any method for computing the estimates for the evaluation of relational algebra operations from the quantitative information on **DB** (relations sizes, attributes selectivity, and so on) may be employed for v^* and e^*. For instance, in our experiments described in the next section, we employ the standard techniques described in [12, 13].

[4] For the sake of presentation, we refer to $\min$ and hence to minimal hypertree decompositions. However, it is easy to see that all the results presented in this paper can be generalized easily to any semiring, possibly changing $\min$, $\mathbb{R}^+$, and $+\infty$.

Clearly, all these powerful weighting functions would be of limited practical applicability, without a polynomial time algorithm for the computation of minimal decompositions. Surprisingly, it turns out that, unlike the traditional (non-weighted) framework, working with normal-form hypertree decompositions, rather than with any kind of bounded-width hypertree decomposition, does matter. Indeed, computing such minimal hypertree decompositions with respect to any tree aggregation function is a tractable problem, while it has been proved that the problem is still NP-hard if the whole class of bounded-width hypertree decomposition is considered. A polynomial time algorithm for this problem, called `minimal-`k`-decomp`, is presented in [41].

4.2 Some Experiments

We implemented the algorithm `cost-`k`-decomp`, which computes a minimal decomposition with respect to the weighting function $cost_{\mathcal{H}(Q)}$ and the class of k-bounded normal-form hypertree decompositions. In this section, we report some results of an ongoing experimental activity on the application of `cost-`k`-decomp` to database query evaluation. A more detailed description and further experiments can be found in the full version of [41], currently available at the hypertree decomposition homepage [40]. Our aim here is just to show that Algorithm `cost-`k`-decomp` may significantly speed-up the evaluation of database queries having structural properties to be exploited. All benchmark queries are executed on the commercial DBMS *Oracle 8.i* by using either Oracle standard query execution method, or the following technique, based on minimal decompositions: the query plans are generated by the algorithm `cost-`k`-decomp` (with k ranging over $(2..5)$), by exploiting the information of the data available from Oracle; the plan execution is then enforced in the DBMS by supplying a suitable translation in terms of views and hints (*NO_MERGE, ORDERED*) to *Oracle 8.i*, which eventually executes the query by its engine, following the desired plan. In both methods, we do not allow indices on database relations, in order to focus just on the less-physical aspects of the optimization task.

We tested the methods with different kinds of queries by varying the hypertree width, the number of query atoms, and the number of variables. Here, we report only the experiments on a set of test queries: we consider again query Q_1 described in Example 3.2, as well as two modifications Q_2 and Q_3, such that Q_2 consists of 8 atoms and 9 distinct variables, and query Q_3 is made of 9 atoms, 12 distinct variables, and 4 output variables. All these queries have width 2. They are evaluated over synthetic data: For each query atom p, we first fix the *size* r_p of the corresponding relation, and we then exploit a random generator that materializes r_p data tuples, by choosing attribute values uniformly at random from a fixed set of possible values. All the experiments were performed on 1600MHz/256MB Pentium IV machine running *Windows XP Professional*. Time measurements for query evaluation in *Oracle 8.i* have been done by using the *SQL Scratchpad* utility. We considered different values for the parameter k. It is worthwhile noting that a higher value of k permits to consider a larger number of hypertree decompositions, and can therefore allow to generate a better plan; but it obviously causes a computational overhead due to a larger search space to be explored by `cost-`k`-decomp`. For the experiments reported in this paper, we chose $k = 3$, which seems empirically a good bound to be used in practice for queries with less than 10 atoms. Figure 5 shows the absolute execution times for Oracle and `cost-`k`-decomp` over a database of 1500 tuples. It can be observed that, on all considered queries, the evaluation of the query plans generated by our approach is significantly faster than the evaluation which exploits the internal query optimization module of *Oracle 8.i*.

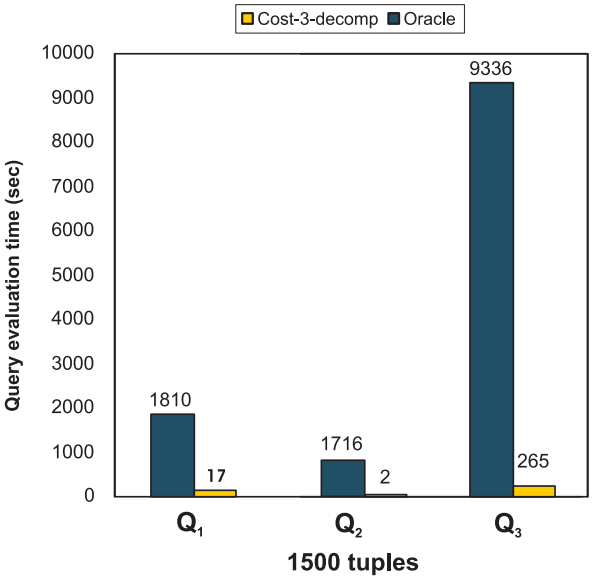


Figure 5: Evaluation time for test queries Q_1, Q_2, and Q_3.

5. CONCLUSION

We described the notion of hypertree width and some of its extensions, and we showed how they can be exploited for identifying and solving efficiently tractable classes of database queries.

Our ongoing work includes an integration of the optimization technique based on minimal decompositions with the query optimizer of the open source DBMS PostgreSQL, as well as a thorough experimentation activity with real queries and databases, loaded with non-random data.

Many interesting questions about structural decompositions are still open and deserve further research. For instance, we do not know if having bounded hypertree width is a necessary condition for a class of queries to be tractable. Moreover, for many real world applications with hundreds of hyperedges, we need good heuristics for computing generalized hypertree decompositions. We refer the interested reader to [22] for a recent graph-theoretic survey on hypertree decompositions, with further results and details on these related issues.

Acknowledgments

The author sincerely thanks Georg Gottlob and Nicola Leone, who worked with him on these issues since 1999, when they jointly defined the notion of hypertree decompositions. Moreover, he thanks Gianluigi Greco for his recent contribution to the weighted extension, and Alfredo Mazzitelli for his valuable work in designing and implementing the tools for experiments.

6. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] S. Abiteboul and O.M. Duschka. Complexity of Answering Queries Using Materialized Views. In *Proc. of PODS'98*, pp. 254–263, Seattle, Washington, 1998.

[3] I. Adler, G. Gottlob, and M. Grohe. Hypertree-Width and Related Hypergraph Invariants. In *Proc. of EuroComb'05*, Berlin, 2005.

[4] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J.ACM*, 30(3):479–513, 1983.

[5] P.A. Bernstein and N. Goodman. The power of natural semijoins. *SIAM J.Comput.*, 10(4):751–771, 1981.

[6] A.K. Chandra and P.M. Merlin. Optimal Implementation of Conjunctive Queries in relational Databases. In *Proc. of STOC'77*, pp.77–90, Boulder, Colorado, USA, 1977.

[7] C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. *TCS*, 239(2):211–229, 2000.

[8] R. Dechter. Constraint Processing. Morgan Kaufmann, 2003.

[9] R. Fagin, A.O. Mendelzon, and J.D. Ullman. A simplified universal relation assumption and its properties. *ACM TODS*, 7(3):343–360, 1982.

[10] J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. *J.ACM*, 49(6):716–752, 2002.

[11] E.C. Freuder. A sufficient condition for backtrack-bounded search. *J.ACM*, 32(4):755–761, 1985.

[12] H. Garcia-Molina, J. Ullman, and J. Widom. *Database system implementation*. Prentice Hall, 2000.

[13] Y.E. Ioannidis. Query Optimization. *The Computer Science and Engineering Handbook*, pp. 1038–1057, 1997.

[14] N. Goodman and O. Shmueli. Tree queries: a simple class of relational queries. *ACM TODS*, 7(4):653–6773, 1982.

[15] G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural CSP decomposition methods. *Artif. Intell.*, 124(2):243–282, 2000.

[16] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Advanced parallel algorithms for processing acyclic conjunctive queries, rules, and constraints. In *Proc. of the Conference on Software Engineering and Knowledge Engineering (SEKE'00)*, pp. 167–176, Chicago, USA, 2000.

[17] G. Gottlob, N. Leone, and F. Scarcello. The complexity of acyclic conjunctive queries. *J.ACM*, 48(3):431–498, 2001.

[18] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions: A survey. In *In Proc. of MFCS'2001*, pp. 37–57, Marianske Lazne, Czech Republic, 2001.

[19] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *JCSS*, 64(3):579–627, 2002.

[20] G. Gottlob, N. Leone, and F. Scarcello. Computing LOGCFL Certificates. *TCS*, 270(1-2):761–777, 2002.

[21] G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *JCSS*, 66(4):775–808, 2003.

[22] G. Gottlob, M. Grohe, N. Musliu, M. Samer, and F. Scarcello. Hypertree Decompositions: Structure, Algorithms, and Applications. In *Proc. of WG'05*, Metz, France, 2005.

[23] E. Grädel. On the Restraining Power of Guards. *J. Symb. Logic*, Vol. 64,pp. 1719–1742, 1999.

[24] M. Grohe. The Complexity of Homomorphism and Constraint Satisfaction Problems Seen from the Other Side. In *Proc. of FOCS'03*, pp. 552–561, Cambridge, MA, USA, 2003.

[25] M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *Proc. of STOC'01*, pp. 657–666, Heraklion, Crete, Greece, 2001.

[26] M. Gyssens, P.G. Jeavons, and D.A. Cohen. Decomposing constraint satisfaction problems using database techniques. *J. of Algorithms*, 66:57–89, 1994.

[27] D.S. Johnson, A Catalog of Complexity Classes, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pp. 67-161, 1990.

[28] P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. *JCSS*, 61(2):302–332, 2000.

[29] T. Korimort. Constraint Satisfaction Problems – Heuristic Decomposition. PhD thesis, Vienna University of Technology, April 2003.

[30] D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1986.

[31] B.J. McMahan, G.Pan, P.Porter, and M.Y. Vardi. Projection Pushing Revisited. In *Proc of EDBT'04*, pp. 441–458, Heraklion, Crete, Greece, 2004.

[32] B. McMahan. Bucket Eliminiation and Hypertree Decompositions. Implementation report, DBAI, TU Vienna, 2004.

[33] C.H. Papadimitriou and M. Yannakakis. On the complexity of database queries. In *Proc. of PODS'97*, pp. 12–19, Tucson, Arizona, USA, 1997.

[34] O. Reingold. Undirected ST-Connectivity in Log-Space, manuscript, 2004, currently available at `http://www.wisdom.weizmann.ac.il/~reingold/publications/sl.ps`

[35] N. Robertson and P.D. Seymour. Graph minors ii. algorithmic aspects of tree width. *J. of Algoritms*, 7:309–322, 1986.

[36] W.L. Ruzzo. Tree-size bounded alternation. *JCSS*, 21:218–235, 1980.

[37] D. Saccà. Closures of database hypergraphs. *J.ACM*, 32(4):774–803, 1985.

[38] P.D. Seymour and R. Thomas. Graph Searching and a Min-Max Theorem for Tree-Width. *J.Comb. Theory B*, 58:22–33, 1993.

[39] F. Scarcello. Answering Queries: Tractable Cases and Optimizations. *Technical Report D2.R3*, Project "Integrazione, Warehousing e Mining di Sorgenti Eterogenee (MURST COFIN-2000)," 2001.

[40] Francesco Scarcello and Alfredo Mazzitelli. The hypertree decompositions homepage, since 2002. `http://wwwinfo.deis.unical.it/~frank/Hypertrees/`

[41] Francesco Scarcello, Gianluigi Greco, and Nicola Leone. Weighted Hypertree Decompositions and Optimal Query Plans. In *Proc. of PODS'04*, pp. 210-221, Paris, 2004.

[42] R.E. Tarjan, and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J.Comput.*, 13(3):566-579, 1984.

[43] J. D. Ullman. *Principles of Database and Knowledge Base Systems*. Computer Science Press, 1989.

[44] A.N. Wilschut, J. Flokstra, and P. M.G. Apers. Parallel evaluation of multi-join queries. In *Proc. of SIGMOD'95*, San Jose, CA, USA, 1995.

[45] J. Van Benthem. Dynamic Bits and Pieces. *ILLC Research Report*, University of Amsterdam, 1997.

[46] M. Vardi. Complexity of relational query languages. In *Proc. of STOC'82*, pp. 137–146, San Francisco, CA, USA, 1982.

[47] M. Vardi. Constraint Satisfaction and Database Theory. *Tutorial at PODS'00*, Dallas, Texas, USA, 2000.

[48] M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. of VLDB'81*, pp. 82–94, Cannes, France, 1981.

[49] C.T. Yu and M.Z. Özsoyoğlu. On determining tree-query membership of a distributed query. *Infor*, 22(3):261–282, 1984.

John Wilkes Speaks Out

on What the DB Community Needs to Know About Storage, How the DB and Storage Communities Can Join Forces and Change the World, and More

by Marianne Winslett

John Wilkes
http://www.hpl.hp.com/personal/john_wilkes

Welcome to this installment of ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today [February 2004] we are at the Department of Computer Science at the University of Illinois at Urbana-Champaign. I have here with me John Wilkes, who is an HP Fellow in the Internet Systems and Storage Laboratory at Hewlett Packard Laboratories in Palo Alto, California, where his research focuses on the design and management of storage systems. John is a member of the editorial board of ACM Transactions on Computer Systems, and until recently he was a member of the Technical Council of the Storage Network Industry Association. John is an ACM Fellow, and his PhD is from the University of Cambridge. So, John, welcome!

Thank you.

John is not a pillar of the database research community; he is a pillar of the ***storage*** *research community, which is next of kin to the database research community. Ironically, though, most database researchers know very little about storage. We tend to treat it as a black box. In fact, John, I can sum up for you everything that most of us database researchers know about disks in six easy terms:* sector, block, seek, rotational latency, transfer rate, *and* RAID*---and that's all we teach about disks in our database courses.*

You forgot *capacity*.

No, we don't do capacity*! We're interested in the number of disk arms, rather than how much you can stuff onto each disk.*

John, you co-authored a wonderful paper last century that described the actual behavior of a particular disk in great detail. That's the March 1994 IEEE Computer *article, "An Introduction to Disk Drive Modeling," that you wrote with with Chris Ruemmler. Can you give us some highlights regarding the actual behavior of real disks today?*

The disk industry has been doing a spectacularly good job of continuing to provide more capacity with lower cost and greater reliability. They have also done a quite good job of increasing the performance of those drives, and they have been slowly making them more sophisticated. Back when we wrote that paper in the early nineties, SCSI disks were becoming the norm, and now they are de facto everywhere. The

complexities, features, and functionalities that the disk drive vendors have put into them slowly increase with time, and the result is that we now get spectacularly good value out of those disks. I am glad I am not in the disk drive business---it is a cut throat environment. But the hard work of the disk industry has left storage system designers and database people in a marvelous position.

I can see how capacity might be important for storage, in the broadest sense. But in the database world, we are more concerned about the number of disk arms and the amount of material under those arms, because performance is so important a factor.

I wish that was more true of the customers who still continue to buy storage based on dollars per gigabyte. Yes, I have agreed with you for a long time that performance is the most important attribute, but not everyone has managed to get to that level of enlightenment.

Are there things that the disk industry could do to really make big improvements there?

They actually are doing it already. You already see a bifurcation in the market between the large capacity, relatively slow drives and the much more expensive, hotter, faster, high speed drives that are optimized for the high end database world. So, the market is driven primarily by what people will pay money for.

When database people think about disks, they don't think about things like the fact that the disk arm first speeds up, then slows down when doing seeks; the big difference between short seeks and long seeks; and all those sorts of details. Should we be thinking about those things, or are those just details that are better ignored?

I've always wondered whether it might be better if the database and storage communities could get together to agree on a slightly more sophisticated model of the storage device behavior.

What features would be in that model?

In the same way that you people may not know about storage devices, we don't know about databases either. But my understanding is that the query models and the optimization algorithms that get used are wonderfully sophisticated algorithms on top of a relatively simplistic base of expectations about how the storage device will behave--

Oh, yes.

--and that seems almost backwards. For example, most people do not put databases on raw disk drives any more; instead, they put them on storage arrays, which have massive amounts of caching and pre-fetching and intelligent algorithms to try and do data layout themselves inside the boxes. And unless you are aware of some of those things, you could get surprised at run time.

I view the creation of a model of storage device behavior as something that neither community could do well by itself. We could probably do much better if we collaborate on trying to find some way of expressing what it is that the storage system is trying to do and what the database would like to have done.

Do storage companies have liaisons with the database companies and--?

Yes, certainly.

So maybe those connections will happen?

Who knows? Part of the problem is that this model and its underlying assumptions need to be brought out in the cold, gray light of day, and I think that is something the research community can contribute to.

So would you say that the research community is not aware of the issues that those liaisons would be looking at?

That is probably too broad an allegation. I have an association with Carnegie Mellon University, where there is research on how to do joint optimizations between the database and storage system. We can probably benefit from more of this kind of cross-community research.

Let's move up a level of abstraction, from the disk to the storage system. Most database graduate students know nothing about storage systems. They do not learn about them in their database classes, and there aren't any classes that focus on storage systems. In fact, there are hardly any professors *who focus on storage systems, yet storage systems are extremely important in industry. Big businesses spend a lot of money buying them. A range of companies build and sell storage systems, from behemoths like IBM through midsize companies like EMC and on down to startups like Network Appliances. In fact, IBM Almaden has about as many researchers working on storage as on databases. Yet there was not even a research conference that focused on storage, until a few years ago when the FAST conference was started. How could an area of such immense economic importance be so invisible in academia, both in research and education?*

I wish I knew. It is absurd. People spend as much money on their storage system as they do on their processors these days, but the amount of attention paid to the two is wildly disparate, at least in academia.

Would you say that the CS curriculum should include coverage of the storage area?

I think it would be a wonderful thing to do. As we mentioned earlier, the performance of many systems is dictated by the behavior of the storage subsystem---and by performance I mean not just an aggregate read and write rate, but reliability and generic quality of service. The storage subsystem is becoming one of the more complicated parts of many systems. Unless we can persuade people that spending time on this area is interesting and worthwhile, I think we will find that we have to rely on what the storage industry will do. It is actually doing a pretty good job, but I am sure we could do better, as we have done in other areas.

What would researchers do that is not already being done in industry?

One of the things that the academic community can accomplish is that to try things that would not appear to make immediate business sense. The industry is dedicated to doing what is going to be delivered next year, and that limits very much the scope of things that can be tried. One of the great things about the academic community is that they can go far enough ahead to say, “Hey, this is worth looking at, and maybe we don't have to have a business justification for investing in it.” Think of them as scouts, looking out in the world and saying, “What if we tried this? What if we removed that restriction?” The academic community has that wonderful degree of freedom that can benefit everybody.

What hot topics do you think people would work on if we had academics looking at storage issues?

I'd like to see more emphasis put on the large scale systems that people have in the real world. All too much energy is spent on the kinds of things you can do with a desktop system, with a single disk inside it running Linux. That is not the way most real databases and large data stores behave.

So what are those large things like? I think our audience doesn't know. What do they consist of?

Think terabytes on the way to petabytes, rather than a few gigabytes. I was having a conversation this morning with one of your colleagues, and he was talking about having large quantities of data. I asked, "So what does large mean to you?" And he said, "Oh, a few gigabytes, at least."

Oh, come on! Obviously not a database colleague.

This was a database colleague.

You're kidding!

He was interested in the algorithms that you get to apply to the data, and the algorithms were large and complicated rather than the data. For a real-world large-scale scenario, just add a few zeros on the end of all the calculations, move to a world where instead of single disk spindles we are now talking thousands of spindles, move to a world where your multiple data centers have to collaborate 24 by 7---they never go down---, move to a world where battery backup and uninterruptible power supplies are the norm rather than the exception. Those are assumptions that change many things. One of the things that has been neglected for many years---and I made one stab at it but there is plenty of work still to be done---is how to guarantee quality of service in that space. I think we need to try to move to a world where predictability is the most important parameter: if it works this way now and I like it, I want to work it that way tomorrow, and I will still like it.

Predictability is a great goal, but I know you also work a lot on making self-tuning storage systems, and in my experience, self-tuning systems tend not to be so predictable because they change their behavior. So how do you reconcile those two goals?

The point of being self-tuning is to achieve some end result which ought to be "the system is doing what I want," i.e., behaving the way I like, and that is a kind of predictability. So I view self-tuning as a way of responding to the vagaries of the environment and workload in order to be able to achieve a target business goal or, in my case, a quality of service goal.

So in that continuum between predictability and performance, you would rather skip the peaks of really excellent performance if it meant you could deliver things at a steady rate?

No, that depends on what you, the user, want. If meeting those peaks is more important to you than meeting the steady rate, you should say so and the system should adjust accordingly.

So, you can tune the relative importance of those two parameters, the consistency and the peak performance. Are there other tunable variables besides those two?

For example, is the system reliable? Does it store data in a way that is not going to get corrupted? Maybe the data is transient and I can make optimizations to get better performance at the cost of things going wrong once in a while. The system should not pre-decide the level of reliability available to the user.

Performance is the easy tunable parameter, the one most people have focused on so far. In some sense, we have made pretty good progress there. Let's work on some of the other areas, such as reliability, availability, usability, functionality, manageability. Very few systems are static and unchanging over their lifetime; instead, they are always being tinkered with, added to. Their requirements are being changed. And the easier it is to make those changes and accommodate those things, the better the resulting system is going to be.

The database community is very interested in having self-tuning database systems. What kind of techniques do you use to make storage systems self-tuning?

Probably much the same techniques, actually. I have long advocated the notion of a control feedback loop, where you take measurements of the system; compare those measurements against what you would like to be happening (which by the way, requires ways of describing what you like); design some kind of response, perhaps diagnosing some kind of problem; determine what is currently the most appropriate solution to apply; and then find some way to apply that solution, preferably in a non-disruptive fashion; get the system back to a state where it is operating again; and repeat the whole process. The only question is, how sophisticated can you be in each of those stages? We started off with really simple measurements of average transfer rates we were trying to achieve. The design considered how far across multiple spindles to spread the data. Now we are moving towards taking into account more things, like reliability. You can imagine the system responding, for example, if something within it breaks. If one thing breaks, that's not such a big deal, but if three things start breaking, maybe you should be rather more aggressive about making sure the reliability goals are being met.

When you see that the system isn't behaving quite the way you wanted, and you want to move it closer, do you rely on a cost model to determine whether the changes you are likely to make are going to move it in the right direction?

There are two different philosophical approaches. One is prediction-based: you put a lot of emphasis on being able to predict the result of a potential change, and you work quite hard to make sure those models are accurate and reliable, so that you can use them to explore potential alternative designs. If your model is right, then when tuning, you will probably get to where you want to be very quickly. My group has done a lot of work in that space, and put a lot of effort into trying to produce good cost models. So that is philosophy number one.

Philosophy number two, which I am now moving towards myself, says that we will never know the system well enough to model it extremely accurately, so let's do a good job of modeling but not bend over too far backwards to try for a perfect prediction. Instead, let’s do a good enough job that when tuning, we can say, “*That* direction is the one I want to go in.” And then perhaps lift some ideas from traditional control theory to determine how far should we go in that direction, and put in a responsive feedback loop.

The issue I have found in my own research is that if I require a very accurate cost model, then every system needs an installation specific cost model, and who is going to create that?

The system should create the cost model itself by learning. Observations are wonderful ways of deducing how the system is behaving. My own group started with our own very well-crafted, hand-tuned analytical models of the likely behavior of the system, and after a while we decided that this is crazy: it is too much work and it requires too well-trained people. So we instead moved to an approach where we don’t measure the heck of out something; instead we build a little table and use that to extrapolate, predict, interpolate. That is the approach we now use. It is more robust, the information is easier to gather, and as the system runs you can just add more entries into the table and make it more accurate for your particular situation. So I believe in this approach----models that get better on their own as you use them.

So you can predict the performance of an expected workload very accurately with a little table?

At least as well as the clever analytical models.

I see. What accuracy can you could attain with that kind of simple table?

We tend to be comfortable once we get to prediction errors of 20-30%, which for one of these complicated storage systems is pretty good. Remember again, with a feedback system, the cost model does not have to be exactly right, it just has to be good enough to send you in the right direction.

So when the storage system is tuning itself, what kind of parameters does it have control over?

The easiest parameter is how many resources get used: disk spindles, memory available in the caches, potential access paths (if there is more than one of them, which there often are for reliability reasons). The second parameter is the way those resources get used. There are lots of policies embedded inside the storage systems: what order requests get serviced in, how long are data left in a write buffer before they get flushed out to the backend storage device, what kind of pre-fetching algorithms should you use, when should you declare that a request pattern is sequential, and how should you place the data on the storage devices. The behavior of the outside of the spindle is different than the inside of the spindle, so the placement of two things on the same spindle can give widely different performance. You could start out with what appeared to be two sequential workloads touching two objects on the same spindle, that when merged are essentially random access with terrible performance for both of them, just because they continue to seek between the two objects. There are actually quite a lot of knobs if you look inside a modern storage device; they are incredibly complicated.

And those knobs will be interacting in nonlinear ways?

Absolutely, just delightful. [Laughs.]

If you have distilled your whole performance model down to a simple table, how can that table capture these nonlinear interactions?

You said *simple table*. I just said *table*.

Oh, what kind of table is it?

It is a multidimensional table. The space that you are operating in has one dimension for each of the knob settings you could have.

So you aren't just measuring, *you are* measuring the heck out of *the system. With all these different combinations of parameter settings, everything is being measured all the time.*

That is our ideal. I mean, why throw this data away? You might as well use it. As time goes by and you get more comfortable with the prediction accuracy, then you can perhaps tone down the enthusiasm with which you gather data and put it into the table. But at first, you may as well take advantage of as much of the data as you can. We have learned that people are not very good at predicting which axes matter.

We had a little piece of work that we never got to publish because two or three people and a summer student worked away for several months trying to built a predictive model for caching behavior on a couple of disk arrays, and we could never get it right. We were off by factors of 2 to 10 in performance behavior. We obviously didn't capture enough of the important magic parameters. The approach of trying to predict everything analytically works well for some things, but for others it has limitations.

Were you trying to model a preexisting storage system, or were you also building the system?

We had two systems we had just bought off the shelf. Standard--

*Shouldn't these models be created by the people who put the devices together and **know** what policies they are using for caching, and so on?*

You would think so, but this is in the business world. Those people haven't that motivation. Vendors provided models in certain places, but the people who had designed the systems thought those models were remarkably simplistic. Vendors sell a lot of disk arrays based on benchmarks, which push the system into extreme behavior on very simplistic loads. We were much more interested in realistic loads. We do a lot of work with trace replay from systems where we have measured the actual behavior, as opposed to synthetic loads. We have discovered that the synthetic loads that people use are not very close to real loads, except for the very special case of benchmarks. But real workloads are nothing like benchmarks.

Another open problem is the workload merging problem. You have one set of work, index accesses or something like that, and another set of table accesses. Now put them together. How they interleave is a very complicated description problem. And the prediction problem for that is even worse.

The sad thing about that is that if you were up at a higher level, maybe at the application level, you would know that information. But at the storage system level, it has been lost, and all you are getting is this worthless little low level trace, from which you are trying to deduce the high level behavior. Why can't you have the higher level tell you this up front, so you don't have to try to guess it?

I think we should ask the database people that. Why wouldn't they pass the information down? Intentions are incredibly valuable piece of information.

How would database people pass that down to you? (I think they should *pass it down, for the record.)*

Let's start really simply: just tell me that this access path that I'm about to do will be random, this one will be sequential. Just actually telling us about sequential accesses before they happen is probably going to be the single most useful thing to do.

One thing that database people have on hand already is the query plans for things that are going to be executed, and they could certainly share that information. I don't know if it would help you or hinder you.

It seems tragic to have all of that useful information gathered and then not passed down.

Thrown away.

To be fair, both communities are guilty, because we have not managed to sit down and negotiate an interface for passing this kind of information down to the storage system. The academic community can help in figuring out what is the right interface. How simple can the interface be, to get the maximum return value?

Also, I think there isn't exactly a normal form for representing query plans---it is more vendor specific. But from a query plan, you could tell which things might be interleaving randomly and which things were actually related, and could probably handle things a lot better.

The other interesting question is, when do you pass this information to the storage system? Clearly at run time when the query is issued, you have a quite a lot of information about what the query is about to do, but when you are doing the system design and provisioning and placing data on the system---that is well before you have actually run the system. Perhaps during the tuning process this information could be made available and could be used.

That is a nice segue into something we were talking about earlier, this notion of how if you are tuning at two different levels, what happens when you put the two together, because they could--

A mess--

--have been fighting each other. The only self-tuning OS I have run on was Windows NT. We were trying to do self-tuning, but we had a hard a time tuning anything because the OS kept changing its behavior.

[Perkily.] It was just trying to help.

Just trying to help, yes. And in the end, just like you said earlier, we would have been happier for the OS to be predictable and slower, rather than faster and inconsistent.

A recurring theme in the computer science community is that the people on top always want the thing underneath to not do anything clever, because they think they are in control and understand what needs to happen. When things start out, I think that's probably right. We made a huge leap forward in the disk community when the disk device driver for Berkeley UNIX came along, because it actually had a model of how the disk drive was going to behave and got hugely better performance. But then the disk drive guys moved on, and people need to let go of assuming they have complete control of the innards of the system. We need to move to a world where the user declares, "I want you to behave this way; I don't need to know how you do it, but the behavior I want is like this"---and then the storage system can optimize around that. We haven't made that transition yet.

In the case of the interface between database and storage systems, how would we specify that behavior? Would it be in terms of quality of service? For what kind of things?

We should specify the behavior that is most important for the database level: reliability, performance, availability, those kinds of things. They all matter, so they all have to be specified somehow. I am being a little vague here about precisely how that is done. We have done some work in that space, but we have not done the validation of putting it into a database and a storage system and having the two collaborate. That is something I would like to see done, as there are a lot of open-ended questions regarding how much better could we make the system if we are willing to exchange information across that boundary.

I don't have a model in my head for what database people would tell the storage system that they want their behavior to be. At the highest level, I could imagine handing a workload to the storage system and saying, "Here, run this and be sure you finish it all in a certain amount of time, with a certain response time, and don't lose the data." I can imagine that, but surely we are talking at a lower level.

I am going to let you know that I am making this up on the fly.

That is okay.

Suppose we pass the query plan down to the storage system and say, "I am going to do all these things in this order." Surely we could do better with that information. Now maybe that is too much information to pass down. The question is how much could we back off from passing down that entire query plan.

But how would you specify your non-functional requirements? Would you say, "Here is my query plan and I want it done in the next five minutes"? Would it be more like a real time system, then?

I think you might do that. I could imagine saying, "Here is the sequence of requests I'm going to emit. Here are the dependencies I have: I cannot issue that request until this one is completed, or this one is completed

plus so many seconds have gone by because I need to process the content. Execute them in any order that makes sense as long as it conforms to this set of dependencies." So that provides degrees of freedom: "You can do anything you like as long as it doesn't violate this constraint." Those degrees of freedom are what give you opportunities for optimization down below.

So the constraints would be perhaps total time spent and consistency constraints for correctness and maybe some constraints having to do with reliability and those other dimensions.

And probably things like the amount of buffer space available. The thing I have at the back of my mind is the experience that we had with disk directed I/O in the parallel systems community, where once upon a time we used to have the people using the data issue low-level requests to the backend storage nodes. After a while we realized that the backend nodes were the bottleneck, and the better thing to do was to tell them what we were trying to accomplish and let them do the sequence on their own.

And let them figure out the sequence on their own, yes.

Same idea now, just applied to the storage system of the database.

I think you may be the first person I have interviewed whose PhD is from a European university. If you were graduating right now, would you still move to the US?

That is a good question. I don't know. When I came over 20 years ago, the opportunities in the United States were much more interesting and attractive than in the UK; I was an operating systems person and the west coast of the US was where things were happening. These days the European computing environment is much more attractive. Cambridge, where I came from, now has little vibrant start-ups, with all sorts of activities around the university; "Silicon Fen" they call it. That provides much more attractive opportunities than existed back then. I now work for HP, and they now have a big research lab in Bristol. That is where I grew up, so it is another potential attractive opportunity. And the climate has changed a lot. It used to be the case that all of the interesting research in hard-core systems was done in the US, and that is less true now.

Do you see a globalization, at least in hard-core systems area research, or is that specific to England? You were mentioning places in England.

The globalization of talent: some people have now chosen to put buildings around where some of the talent is, as opposed to moving the talent to the buildings.

Recent years have seen some hard times for industrial research labs. What do you think the future holds for industrial labs? Will they become extinct?

I certainly hope not. And I doubt it. This is one of those business questions of how to choose to invest in your future. There are many different models, all of which appear to be viable. At HP, we have chosen a model where we have a core center research lab whose funding is protected from the day to day throes of getting products out the door. Other companies have chosen different ways, and they have been more or less successful depending on many different factors. But insofar as HP Labs --and other research labs-- continue to provide value to their investors, they will continue to exist.

It all comes down to that definition of value. So what is the value that you are providing at HP and that people can't live without?

Ink jet printing is the one we always point to.

And what year did you do that?

It is a while back. We need to do a few of those big contributions each decade. It is hard to do that, I mean, we are there to take the big risks. I used to have this conversation with the head of HP Labs, saying that if we are not failing in two-thirds of the things we try, then we are not taking enough risks.

So what have you done since inventing ink jet printing?

PA-RISC, IA64 Itanium, and a bunch of other things that are smaller activities and that do not show up so much. But there are 600 or so people now at HP Labs leading the way---

I see. But you did not mention anything in storage.

[Laughs.] I was trying to paint the broader picture.

Okay. Have any special things at HP come out of your storage group?

I like to think that we helped AutoRAID get out the door, which was one of HP's first smart disk arrays. HP has been quite big in the storage management arena for quite a few years, and we have had influence on the storage management plan. Have we had as much impact as I would have liked? Absolutely not. We can always aim for more. I am currently moving into the adaptive utility computing space, and I would argue that the research we have been doing in my group for the past five or six years has been right on target. We just happened to look at the storage domain first. As a proof of concept and a demonstration of what can be done with kinds of control feedback loops we were talking about earlier, the storage domain has been great. HP is now shipping products that provide these kinds of facilities, partly I think because we helped lay the groundwork showing that these things are possible and can be done.

Do you have any words of advice for fledging or mid-career database researchers or practitioners?

If you are not in this for fun or profit, what the heck are you doing here? Think about what you are trying to accomplish! The failing that I see is the people who get enamored of what I would call "science projects": little simple things that they would like to try to take to fruition and get across that published paper barrier. Yes, that is fine. We all have to do that, and occasionally it is wonderfully rewarding to take a simple idea and move it. But I would encourage people to take a step back and say, "Well that is great, but who cares? What else do they care about? How can I solve their *other* problems, rather than just assuming that this is a good solution?" Go back and see if you can find a broader set of questions.

If you magically had enough time to do one more thing at work you are not doing now, what would it be?

[Laughs.] Sleep.

Oh, you are going to sleep at your desk, are you?

[Laughs.]

That *will look good.*

Magically. You said magically. Okay: create more hours in the day.

But weren't you saying last night that the life of us academics is way too hard and we work so *hard, and here you are the next morning saying that* you *need more sleep!*

I like to think of it this way: you get to do all the things I get to do, plus you have to teach, plus you have to go through grant proposal writing. The reason I work hard is that I enjoy it, so that is not an issue.

If you could change one thing about yourself as a computer scientist, what would it be?

I would get better at persuading other people that the things I think are interesting are interesting to them, because what we accomplish---

Marketing.

No, that's unkind!

Unkind? I think it-- [Laughs.]

I think marketing is trying to persuade people to do things that they don't want to do.

Not necessarily. When you are showing the importance of what you are doing, such as when you are writing a grant proposal, I consider it to be marketing, but it is not an evil thing (if you think marketing is evil).

Okay. In that sense, the positive sense, it is not evil.

[Teasing,] [To camera man.] We have got that on the audio tape, don't we?

What you really accomplish, you cannot accomplish alone; you have to do it through and with other people.

True, absolutely.

And getting them as excited as you are---and it feeds on itself, right? If other people get excited, you do too.

Do you know how you could really build that skill and give back to the community? You are not going to like this; are you ready?

I can see something coming. I am being set up.

What you (and anyone out there who is like you) need to do is go to a funding agency and start a program in an area you think is important. In your case, that area might be storage, because the storage research funding out there now is tucked inside other programs--- they don't stand on their own. That is how you could both build that skill (because program managers have to convince the upper levels that their topic is important, and you know that storage is incredibly important, so it should be an easy sell) and then get that funding program in place, which would encourage more academic types to start looking at storage issues.

That is an excellent idea, but I may not be the right person for it.

Well, I hope someone reading this article is the right person.

That would be wonderful.

Thank you, John.

My pleasure. Thank you.

Reminiscences on Influential Papers

Kenneth A. Ross, editor

See `http://www.acm.org/sigmod/record/author.html` for submission guidelines.

Graham Cormode, Lucent Bell Laboratories, `graham@dimacs.rutgers.edu`.

[Noga Alon, Yossi Matias and Mario Szegedy: The Space Complexity of Approximating the Frequency Moments. Journal of Computer and System Sciences 58(1), 137–147, 1999. Conference version appeared in STOC 1996.]

This paper recently won the theory community's prestigious Goedel award, yet it has also been tremendously influential in the the database community in general and at a personal level on my research. In one short paper, the authors established many founding principles of the data stream model, gave ingenious algorithms for computing the second frequency moment, a general method for finding all frequency moments, and lower bounds on the space needed. While these problems initially appear abstract, the paper has had wide influence on algorithms, database and network research.

Reading this paper as a graduate student was a revelation: having been thinking about some related questions before reading the paper, I found some of the results almost unbelievable. The crisp presentation inspired me to learn the relevant tools used to give such initially surprising proofs. For the database community, the best known result from this paper is a simple randomized "sketch" data structure which neatly and simply computes F_2 (sum of squares) of a stream of values that can be arrive in a very general update model. Subsequently, it has been shown that this summary can also be used to compute point estimates, range sums, inner products (join size estimates) and more. All this from just one page of the original conference paper! Indeed, one of the inspirational aspects of the paper is that it contains not just one excellent idea but many excellent ideas. Further, the fact that the paper appeared in a theory conference reminds us that researchers are not confined to a single area, but can work across boundaries, and read and publish wherever seems most appropriate.

Amol Deshpande, University of Maryland, `amol@cs.umd.edu`.

[Ron Avnur, Joseph M. Hellerstein. Eddies: Continuously Adaptive Query Processing. Proceedings of the 2000 ACM SIGMOD Conference, May 16-18, 2000, 261–272.]

I was initially hesitant to write about a relatively new paper written by my graduate advisor, but no other paper has quite influenced my research career as much as this original *eddies* paper. This paper was published when there was an increasing concern about the applicability of traditional query optimization techniques in domains such as static web sites, dynamic web services, and data streams, and various *adaptive query processing* techniques were being introduced. This paper presented probably the most radical overhaul of the query processing architecture, proposing use of a new operator called an *eddy* (the work was done in the context of the *River* data flow system, hence the name), through which all tuples processed by the system are routed; the eddy reacts to observed data characteristics by changing the order in which tuples are routed through the remaining operators, thus effectively changing the query plan used to execute the query.

This paper was fairly controversial when it first appeared, with many concerns being raised about its efficiency. The technique did form the basis of the Telegraph system, and its impact can also be seen in other follow-on work in adaptive query processing. I am personally biased, and won't discuss the merits of the idea itself in this note.

Although the jury may still be out on the long-term impact of this paper, there is no doubt that it significantly influenced my own research in many ways. To be honest, even though there was much work going on in our group on eddies, it wasn't until much later that I became interested in it. Because of its lack of rigorous analysis, this paper confuses the relevant, impactful concepts presented from those that were ad-hoc or had no merit. For example, it never proves under what conditions an eddy will produce correct results (this is not obvious). But once I started analyzing the technique and tried to put it in perspective, I was amazed by its simplicity and beauty, and very surprised that something so innovative could be discovered in a well-established area. Other than its direct influence on my research, this paper also changed the way I think about query processing, and taught me to think twice about accepting even well-established doctrines. In retrospect, it was the perfect paper for me, then a relatively immature graduate student looking for research ideas, to have run into.

Panagiotis Ipeirotis, New York University, panos@stern.nyu.edu.

[David R. Cox: Regression models and life-tables. Journal of the Royal Statistical Society, Series B 1972; 34:187-220.]

I read this paper while working on the problem of updating word- frequency histograms that characterize a web database. Since databases change over time, the associated statistics should also be updated to reflect content changes. The fundamental challenge in this problem is to learn how long it takes for a database to "change." Knowing this time, it is then possible to schedule updates appropriately. Most of the solutions for related problems in the database field either assumed that this time is given, or that we can learn it by observing a database for long periods of time: a costly solution when dealing with web databases. Then I realized that actuaries face a similar problem all the time: predicting the lifetime of insured clients. I started checking the literature, and this paper by Cox was by far the most frequently cited reference. I started reading the paper, realizing that statisticians have been developing solutions for the problems that I faced, for more than three decades. While it took me some time to read and digest the proposed methods, at the end I was impressed. The proposed regression model was conceptually simple, theoretically sound, and made a very limited number of assumptions about the data. The model could effectively use incomplete ("censored") data and could deal with non-linear effects. Furthermore, the applications seemed (and are) endless. However, even after reading the paper, I was afraid that such a "traditional" method from statistics could only be applied to relatively small number of data points, and that it would not scale for the large data sets. I could not be more wrong. The regression ran quickly, returning simple and elegant formulas that I could subsequently use for scheduling updates.

This paper, and subsequent work in the survival analysis field made me realize that "no man is an island." Research from other fields can be easily applied to traditional database problems. I am still working on similar problems, and the work by Cox and other statisticians is a very fertile source of methods for my research. I strongly believe that anyone who works on similar problems (view maintenance, histogram refreshing, updating cached data, publish/subscription techniques, and so on) should read this paper and be familiar with the general field of survival analysis. The 8,000 citations returned by Google Scholar for this paper just prove its importance.

Donald Kossmann, ETH Zurich, kossmann@inf.ethz.ch.

[Michael J. Carey, David J. DeWitt, Jeffrey F. Naughton: The OO7 Benchmark. SIGMOD Conference 1993: 12–21.]

I have a personal and more general perspective on this paper. From my personal perspective, this paper saved my PhD thesis and so it had a huge impact on my career. In my thesis, I was working on "pointer swizzling" and "dual buffering". I strongly believed in my ideas, but I had no way to show that they were

worth-while using the existing benchmarks. It turned out that my ideas worked extremely well on the OO7 Benchmark. I was lucky.

In general, benchmarks shape a field and strongly impact the research directions pursued. This was definitely true for OO7 because in the OODB research community everybody tried to win the OO7 battle. Today, we know that OODBs were a (commercial) failure and research on OODBs has stopped. Now, what does that mean for OO7? There are two standpoints: (a) OO7 was harmful because it made us work on the wrong problems and, as a result, we could never get the glorious ideas of OODBs to work; (b) OO7 was very helpful because it made us realize early what the potential killer-applications of OODB technology would be and that OODBs were doomed to fail. No matter what standpoint you take, OO7 was clearly extremely influential: to the better or to the worse — I cannot judge.

Lucian Popa, IBM Almaden Research Center, `lucian2@us.ibm.com`.

[Catriel Beeri and Moshe Y. Vardi: A Proof Procedure for Data Dependencies. JACM 31(4), October 1984, 718–741.]

This is one of the classical papers of relational database theory. I first read the paper as a graduate student at the University of Pennsylvania. It was my advisor, Val Tannen, who realized that our work on equational rewriting of queries under constraints had something to do with the chase. He asked me to read several papers on the subject, including the Beeri and Vardi paper. I took several weeks to read the paper. The notation is a bit outdated and it may take a while to get used to it, but underneath, the paper is a gem. It is probably the single paper that influenced my research the most, as I learned two fundamental concepts that afterwards I used in many occasions and I still use: 1) the class of tgds and egds to express data dependencies, and 2) the chase with tgds and egds.

The paper introduces tgds (standing for tuple-generating dependencies) and egds (standing for equality-generating dependencies) as a general formalism to include most prior forms of database dependencies: inclusion dependencies, foreign key constraints, multivalued and join dependencies, functional dependencies, key constraints, and more. There were two other independently developed formalisms that turned out to be equivalent to tgds and egds (the embedded implicational dependencies of Fagin and the algebraic dependencies of Yannakakis and Papadimitriou). However, Beeri and Vardi are the first to define the chase at this level of generality. The paper goes into great depth to prove the important properties of the chase, such as completeness. In addition to the main results, many of the technical lemmas, the proof techniques as well as the various subclasses of tgds and egds that appear in the paper are still relevant and instructive today.

Although the formalism of tgds and egds did not quite make it as a practical framework for database dependencies (the more common foreign key and key constraints are simpler to handle and maintain), today such dependencies are striking back and not necessarily as database constraints. Tgds and egds are at their best use when describing relationships between schemas or between components of schemas. They can describe relationships between physical database structures and logical schemas. Such description can then be used together with the chase and other techniques to perform query reformulation. Tgds and egds can also be used to describe data integration style of schema mappings, in which case the chase can be used to give an elegant semantics for the process of transforming data from one schema into another. This is in fact the formal basis for the schema mapping language developed at IBM for the Clio data exchange system. So, the language of tgds and egds and their chase are still very relevant today, after more than two decades.

CMM and *TODS*

Richard Snodgrass

rts@cs.arizona.edu

The *Capability Maturity Model* [4] is an orderly way for organizations to determine the capabilities of their current processes for developing software and to establish priorities for improvement [2]. It defines five levels of progressively more mature process capability [3].

Level 1: Initial The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.

Level 2: Repeatable Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

Level 3: Defined The software process for both management and engineering activities is documented, standardized, and integrated into an organization-wide software process. All projects use a documented and approved version of the organization's process for developing and maintaining software. This level includes all the characteristics defined for level 2.

Level 4: Managed Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures. This level includes all the characteristics defined for level 3.

Level 5: Optimizing Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies. This level includes all the characteristics defined for level 4.

You may be asking, what does a maturity model for software development have to do with databases generally and with *TODS* in particular? Well, CMM has been applied to personnel management, quality management, and even weapons system development. And it can be used as a framework for evaluating the journal review process, as we will do here.

Manuscript review at *TODS* started, logically, at Level 1. In 2001, the ACM Publications Board approved a broad policy [1, 5, 6] that raised publishing of ACM journals and transactions to Level 2. In 2003 ACM adopted the Manuscript Central[1] web-based manuscript tracking system [7], raising its manuscript reviewing process to Level 3.

In parallel with these efforts at the ACM Publications Board level, I have been refining the reviewing process for *TODS*. In October 2003 I released the first edition of the ***ACM TODS* Associate Editor Manual**, with revisions in April 2004 and October 2004. This manual, at 22 pages, is quite detailed.

I have also been collecting detailed statistics since July 2001. Some of these statistics are reported on the *TODS* web site[2]: turnaround time, article length, number of articles, and end-to-end time [6]. I have also kept records on the turnaround time of individual Associate Editors, and have closely monitored the progress of individual papers.

[1] http://acm.manuscriptcentral.com

[2] http://www.acm.org/tods/TurnaroundTime.html

Through these efforts, and through a series of internal policies regarding the reviewing process that has been adopted by the *TODS* Editorial Board, all of the statistics has improved, some considerably [8]. Average turnaround time is now down to 13 weeks, average article length has been brought down to levels last seen in the mid-1990s (under 40 pages), the number of articles per volume is back up to that last experienced in the early 1990's (21 articles per year), and average end-to-end time is down to 17 months, last experienced in the 1970's.

The result is that *TODS* is now operating at CMM Level 5.

Why should you, dear reader, care about internal processes at *TODS*? The short answer is that by being at Level 5, *TODS* can provide assurances as to how *your* submission will be handled.

Average turnaround and end-to-end times are nice, but what authors really care about is how long *their* submission will take to be reviewed. Addressing this concern involves both average and *maximum* times. A low average turnaround time is of little reassurance to someone experiencing an abnormally long turnaround time. As an example, while the average turnaround time for papers submitted in January 2002 to *TODS* was a quite reasonable 5.5 months, one paper submitted that month had to wait almost nine (!) months for a decision.

By virtue of being at CMM Level 5, the variance of the turnaround time could be monitored and improved, as shown in Figure 1.

The turnaround time has been slowly decreasing over the past four years. This figure shows four sets of data. The bottom line is the *average turnaround time*, a moving average of the turnaround time for papers submitted in the indicated month. To smooth monthly variations, the moving average includes all of the submissions for the previous year. Each data point represents dozens of papers. The value for January 2005, 12.5 weeks, is the average turnaround time for all of the papers submitted between (inclusive) February 2004 and January 2005.

The next line up is the average turnaround time for external reviews only, a moving average of the turnaround time for papers submitted in the indicated month. This includes only submissions that went out to external reviewers and specifically excludes desk rejects. The value for January 2005, 15.6 weeks, is the average turnaround time for external reviews of all the papers submitted during the year up through January 2005.

The points, one per month, denote the maximum or peak turnaround time for submissions in the indicated month. Each point represents a single, unusually slow paper submitted during the indicated month. For all the papers submitted in January 2005, the longest turnaround time was 4.9 months (21 weeks).

In terms of turnaround time, *TODS* at 12.5 weeks is now equivalent to conferences (as exemplified by SIGMOD and PODS at 12 weeks), while being more flexible in not imposing a submission deadline.

The straight line is the *committed maximum turnaround time*, the boundary that the Editorial Board has committed to not exceed, for any submission. Several years ago the Editorial Board established a formal policy stating its commitment to providing an editorial decision within 6 months [8]. *TODS* thus joined conferences in guaranteeing a stated turnaround time.

Due to the rigorous application of CMM Level 5, of continuous process improvement as exemplified by the steady lowering of average turnaround time and the compression of the variance in turnaround time by a factor of two, I can announce that the Editorial Board is now committed to providing an editorial decision within *five* months, starting with submissions in 2004. As depicted in the figure., we have met this stated commitment for the past thirteen months. As of the writing of this column (June 29, 2005), all manuscripts submitted before February 1 of this year have been processed and editorial decisions rendered.

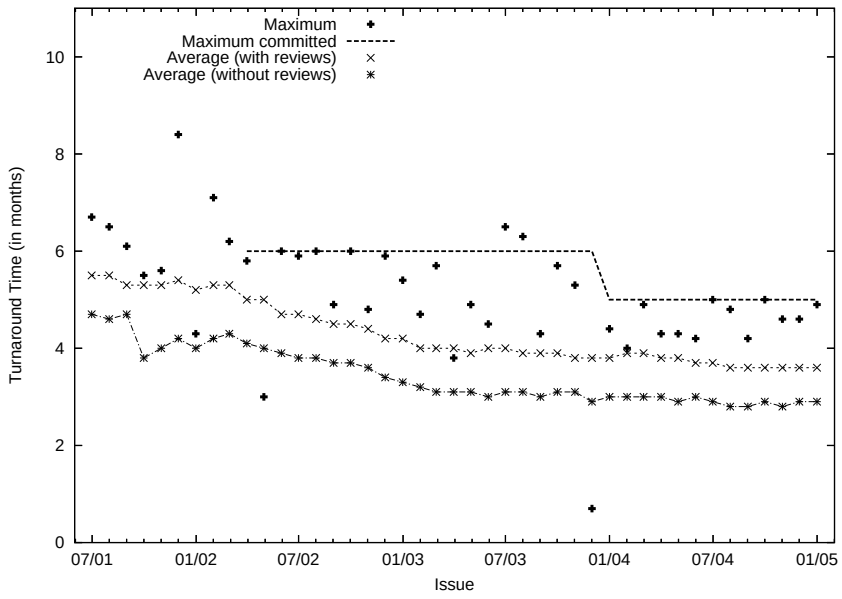


Figure 1: ACM *TODS* Turnaround Time

That *TODS* now matches conferences in terms of turnaround time is a testament to the hard work of two groups of people: reviewers and the editorial board. I will recognize the reviewers in a future column, but here I wish to thank the following people, who comprise the *TODS* Editorial Board, for their dedicated effort work in achieving very fast decisions while upholding very high standards.

Surajit Chaudhuri, Microsoft Research
Jan Chomicki, SUNY Buffalo
Mary Fernandez, AT&T Labs
Michael Franklin, Univ. of California at Berkeley
Luis Gravano, Columbia University
Ralf Hartmut Güting, Fernuniversität Hagen
Richard Hull, Bell Labs
Christian S. Jensen, Aalborg University
Hank Korth, Lehigh University
Donald Kossmann, ETH Zurich
Heikki Mannila, University of Helsinki
Z. Meral Özsoyoğlu, Case Western Reserve
Raghu Ramakrishnan, University of Wisconsin
Arnie Rosenthal, MITRE
Betty Salzberg, Northeastern University
Sunita Sarawagi, IIT Bombay
Dan Suciu, University of Washington
Jennifer Widom, Stanford University

These 18 people are providing a truly valuable service to readers, to authors, and to reviewers. When you see these people, please thank them personally for their role in achieving quick reviews of submitted papers.

References

[1] ACM Publications Board, "Rights and Responsibilities in ACM Publishing," approved June 27, 2001. (`http://www.acm.org/pubs/rights.html`)

[2] Watts S. Humphrey, **A Discipline for Software Engineering**, Addison-Wesley, 1995.

[3] M. C. Paulk, Bill Curtis, and M. B. Chrisis, "Capability Maturity Model for Software, Version 1.1," Software Engineering Institute Technical Report, CMU/SEI-93-TR, February 24, 1993.

[4] SEI, **The Capability Maturity Model: Guidelines for Improving the Software Process**, Software Engineering Inst. Carnegie Mellon Univ., Addison-Wesley, 1995.

[5] Richard T. Snodgrass, "Rights and Responsibilities in ACM Publishing," *CACM* 45(2): 97–101, February 2002.

[6] Richard T. Snodgrass, "Rights of *TODS* Readers, Authors and Reviewers," *SIGMOD Record*, 31(4):5–9, December 2002.

[7] Richard T. Snodgrass, "ACM *TODS* in this Internet Age," *SIGMOD Record*, 32(1):4–5, March 2003.

[8] Richard T. Snodgrass, "Journal Relevance," *SIGMOD Record*, 31(3):11–15, September 2003.