Linköping Studies in Science and Technology

Thesis No. 1427

Supporting Scientific Collaboration through Workflows and Provenance

by

Tommy Ellqvist

Submitted to Linköping Institute of Technology at Linköping University in partial fulfilment of the requirements for degree of Licentiate of Engineering

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 2010

Supporting Scientific Collaboration through Workflows and Provenance

by

Tommy Ellqvist

January 2010
ISBN 978-91-7393-461-9
Linköping Studies in Science and Technology
Thesis No. 1427
ISSN 0280–7971
LiU–Tek–Lic–2009:35

ABSTRACT

Science is changing. Computers, high-speed communication, and new technologies have created new ways of conducting research. Researchers from different disciplines are processing and analyzing increasingly large volumes of scientific data. This kind of research requires that the scientists have access to tools that can handle large amounts of data, enable access to vast computational resources, and support the collaboration of large teams of scientists. This thesis focuses on tools that help support scientific collaboration.

Workflows and provenance have proven useful in supporting scientific collaboration. Workflows provide a formal specification of processes for scientific experiments, and provenance offers a model for documenting data and process dependencies. Together, they enable the creation of tools that can support collaboration through the life-cycle of scientific experiments, from specification of processes for scientific experiments to validation of results. However, existing models for workflows and provenance are often specific to particular tasks and tools. This makes it difficult to analyze the history of data that has been generated over several application areas by different tools. Moreover, designing workflows is time-consuming, often requires extensive knowledge of the tools involved, and may require collaboration with researchers with different expertise. This thesis addresses these problems.

Our first contribution is a study of the differences between two approaches to interoperability between provenance models: direct data conversion and mediation. We perform a case study where we integrate three different provenance models using the mediation approach, and show the advantages compared to data conversion. Our second contribution serves to support workflow design by allowing multiple users to concurrently design workflows. Current workflow tools do not allow users to work simultaneously on the same workflow. We propose a method that uses the provenance of workflow evolution to enable real-time collaborative design of workflows. Our third contribution considers supporting workflow design by reusing existing workflows. Workflow collections for reuse are available, but more efficient methods for generating summaries of search results are needed. We explore new summarization strategies that consider the workflow structure.

This work has been supported by CUGS (National Graduate School in Computer Science).

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Acknowledgements

The research work presented in this thesis was made in collaboration with researchers at the Scientific Computing and Imaging Institute (SCI) and the School of Computing at the University of Utah. Together, we have explored many interesting research problems, and I am very grateful to be part of this work.

This thesis would not have been possible without the help of my three supervisors. You have my deepest gratitude.

Associate Professor Juliana Freire has always been encouraging, providing feedback on my work and many great ideas. She gave me the opportunity to visit and work at the University of Utah, which was a great experience for me.

Professor Nahid Shahmehri introduced me to research in computer science. Through her support and encouragement I have learned much about what it is to be a researcher.

Associate Professor Lena Strömbäck always followed up on my progress and kept me on track. She has always provided me with useful comments on my research, which has helped me become a better researcher.

I thank my colleagues at ADIT (Division for Database and Information Techniques). They have always been helpful, and I have had the opportunity to participate in many interesting discussions, fun activities, and informative lunches. I am grateful to be a student in CUGS (The Swedish National Graduate School in Computer Science). Its courses and events have allowed me to meet many interesting people. Also thanks to Brittany Shahmehri and Mikael Åsberg for proofreading this thesis.

Finally, I want to thank my family. They have always approved of my choices in life and supported me. Especially thanks to Maren, my fiancée, for her love and support during many long days of work.

Tommy Ellqvist
November 2009

Contents

Chapter 1

Introduction

The focus of this thesis is on improving scientific collaboration through the use of workflows and provenance. This chapter introduces and summarizes this work.

1.1 Motivation

The main goal for this thesis is to provide infrastructure that supports and promotes scientific collaboration. Recently, the need for scientists to collaborate in large and data-intensive scientific projects has become increasingly important. Large scale science projects processes and analyzes large amounts of data that involve large teams of cooperating researchers. In order to cope with this, new infrastructure needs to be developed [SPG05]. This thesis builds on two concepts that are important to scientific collaboration: *workflows* and *provenance*.

A *workflow* is a description of a complex process and contains a set of tasks together with their control and data dependencies. A workflow assembles a set of tasks into more complex tasks. A workflow that describes a scientific experiment is called a *scientific workflow*. Scientific workflows can be designed in a workflow editor, executed in a workflow system, and later re-edited, re-executed, shared, and reused. These properties make them suitable for use in scientific collaboration [GDE+07b, KSC+08, SKV+07, XM07, GSLG05]. Workflows contain process information that is a valuable resource for designing and modifying scientific experiments. Workflows can be complex to assemble — it takes time to learn how to compose different components. By sharing workflows, users can leverage each other's knowledge, and learn by example [GSLG05]. Tools are needed that can enable collaborative workflow design. The contributions in this thesis provide a way to collaboratively design workflows synchronously in real time, and also offline, through searching a workflow repository.

Provenance is another key technology in data-intensive research. It provides a model for documenting causal relationships between immutable items. Specifically, *data provenance* enables the preservation of data dependencies, like which input was used to generate a specific result. This makes provenance an essential technology for managing and validating large amounts of data [FKSS08]. A key issue is the lack of interoperable provenance standards and tools. It is hard to develop general tools and combine data generated by different workflow systems. One contribution of this thesis is a method for integrating different models of provenance, and a query API that supports a general provenance model.

In their essence, workflows and provenance represent different aspects of the same thing [CFH$^+$08]. A workflow represents a specification of a process that can be performed in the future, whereas provenance represents processes that have already been performed. In this regard, workflows can be considered to be *prospective* provenance. However, this fact is also their key difference, and is what makes them useful in different parts of the scientific process. Workflows are usually used during design and execution of scientific experiments, whereas provenance is used in validation and presentation of results. Together, they are able to support the scientific process all the way from design to result.

1.2 Problem Definition

In this thesis, we adress the following research question:

- "How can workflows and provenance be used to support scientific collaboration?"

Thus, the goal of this thesis is to "contribute to the improvement of scientific collaboration through the use of workflows and provenance". This thesis addresses three specific topics that serve to answer the research question: data provenance interoperability, collaborative design of workflows, and summarizing workflow search results. Workflows and provenance share many characteristics, but in this thesis we focus on them one at a time. For workflows, we focus on the reusability of workflows that serve to promote scientific collaboration. For provenance we focus on the interoperability between different provenance models. The next section describes the actual contributions.

1.3 Contributions

Our main contributions are summarized below:

- We propose a mediation-based approach to achieve provenance interoperability. We developed a global schema and query API and performed a case study of three different models.

- We explore a novel mechanism which leverages provenance of workflow evolution to support the collaborative design of workflows in a synchronous fashion.
- We study strategies for summarizing workflows in search results, in order to better support the reuse of knowledge in workflow specifications.

1.4 Outline

The remainder of this document is organized as follows:
Chapter 2 describes the problem of scientific collaboration, the vision for the future, the technologies used, and the approach used to reach this vision.
Chapter 3 describes a data provenance integration approach that uses mediation instead of data conversion. This work has been published in the proceedings of SERVICES/SWF 2009 [EKF$^+$09].
Chapter 4 proposes a method for collaborative design of workflows. This work has been published in the proceedings of IPAW 2008 [EKA$^+$08].
Chapter 5 presents a comparison of strategies for summarizing workflows. The work on search result snippets has been published in the proceedings of SIGMOD/KEYS 2009 [ESLF09].
Chapter 6 discusses directions for future work and concludes the paper.

Chapter 2

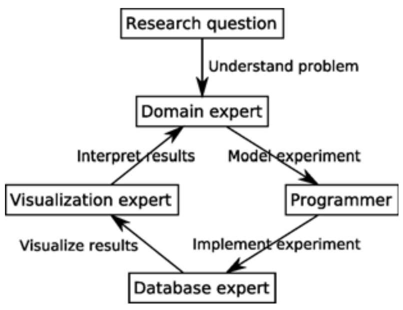
Scientific Collaboration through Workflows and Provenance

There are many aspects of scientific collaboration, such as collaboration patterns in different communities [LPS92] and different methodologies [Bea01]. The focus of this thesis is on the tools used to support scientific collaboration. Workflows and provenance are the two central tools in this respect. The reason is that data manipulation, storage, and logging of scientific experiments are essential to large scale computer supported science such as e-science [HT05], Workflows represent the specification and execution of scientific experiments, and provenance represent the logging of experimental results and data manipulations. Collaborative tools built on these concepts should support the exchange of workflows, data items, data history logs, processes, and experiments.

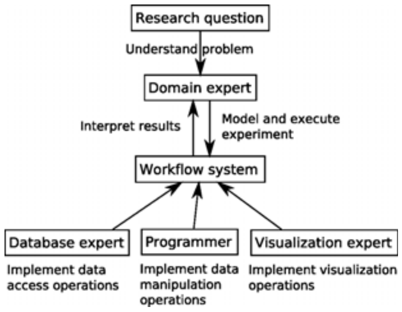
This chapter presents a vision of scientific collaboration. It explains how workflows and provenance are central to scientific collaboration, what issues remain, and the research strategy for addressing these issues.

2.1 Introduction

Science is changing. Computers have become an indispensable tool for many scientists and the amount of available scientific data increases at an exponential rate [HT03]. A key technology for supporting this is *data grids* [CFK$^+$00]. The term data grids describes technologies that provide infrastructure for data storage and processing. They provide services such as storage, replication, access, batch processing, parallelization, security protocols, and scheduling of large amounts of data. These technologies are necessary to handle the vast amounts of data generated by scientific exper-



(a) A scientific collaboration scenario where experiments are set-up and executed in a serial fashion.


Research question
Understand problem
Domain expert
Interpret results
Model and execute experiment
Workflow system
Database expert
Programmer
Visualization expert
Implement data access operations
Implement data manipulation operations
Implement visualization operations

(b) A scientific collaboration scenario including a workflow component to speed up iteration of experiments.

Figure 2.1: Processing large amounts of data requires collaboration between domains.

iments.

The analysis of scientific data in data grids often requires teams of researchers from different domains working together to capture, process, visualize, and interpret scientific data [Fos03]. This can include database experts, programmers, visualization experts, and domain experts. One such scenario is shown in Figure 2.1(a). The domain experts work with the programmers on how to process the data. The programmers work with the database experts to implement the operations and process the data. Visualization experts present the results which are then analyzed by domain experts. This is a slow and tedious process, not only because each iteration takes time, but because science is a process of trial and error, and requires many iterations to achieve the desired objective [FSC+06].

Workflows allows this process to be improved. The idea of the workflow is to describe the whole data processing pipeline using a graph model that expresses data and process dependencies. The nodes in this graph constitute data processing operations, such as fetching and visualizing the data, while the edges describe how data flow between operations. The domain expert can use the workflow to control all the steps in the process, as shown in Figure 2.1(b). First one can specify the input data, then the operations to be performed on this data, then the results can be fed to a visualization operation. By executing this workflow the data is automatically generated and the results are available to the user. There is, however, additional work associated with making the different types of components available to the workflow: All components need to be expressed as modules with clearly defined inputs and outputs [Aßm03]. The advantage comes when the domain expert wants to modify or repeat an experiment: The relevant parameter changes is made and the workflow is re-executed in two simple steps. This approach has a great advantage since the trial and error approach is commonly used in practice [FSC+06]. Much time can be saved as more control is given to the domain expert, who does not need to learn a programming language or how to edit ad-hoc scripts.

Another advantage of using workflows is that it is easy to record *data provenance* [SPG05]. Data provenance describes the causal relationships between data items in the form of a graph model that is similar to the workflow. Its purpose is to record dependencies between data and associated information such as the processes involved, time and date, author, and properties of the experiment. The data provenance graph may span multiple workflow systems and even contain additional steps, such as external data modifications. Data provenance is used to validate, trace, correct, and recreate scientific data.

Workflows and data provenance together facilitate scientific collaboration through common process models and a data validation framework. To fully understand their advantages we will now present an in-depth description of these concepts, and present the current issues in scientific collaboration.

2.2 Workflows

A *workflow* is a description of a complex process and contains a set of tasks together with their control and data dependencies. A *scientific workflow* is a workflow that describes a scientific experiment. Scientific workflow and workflow-based systems have emerged as an alternative to simpler and more commonly used scripting approaches [BL97] for documenting computational experiments and design complex processes. They provide a simple programming model whereby a sequence of tasks (or modules) is composed by connecting the outputs of one task to the inputs of another. This is often preferable to using ad-hoc scripts for repetitive tasks commonly found in scientific research. Workflows can be viewed as graphs, where nodes represent modules and edges capture the flow of data between the processes.

The remainder of this section describes the workflow concept. Section 2.2.1 presents a formal definition of the workflow concept and Section 2.2.2 describes different workflow applications.

2.2.1 Definition of Workflow

The workflow definition used in this thesis is based on a study of several workflow systems [Vis, Tav, GJM+06] and represents core features that are present in all of them.

A *workflow*, as depicted in Figure 2.2, is a set of partially ordered modules whose inputs include both static parameters and the results of earlier computations. A *parameter* represents a data value. A *module* m is an atomic computation $m : P_I \rightarrow P_O$, that takes as input a set of arguments (input ports P_I) and produces a set of outputs (output ports P_O). The parameters are predefined values for ports on a module and can be represented as a tuple $(module\_id, port, value)$. In addition, *connections* link modules

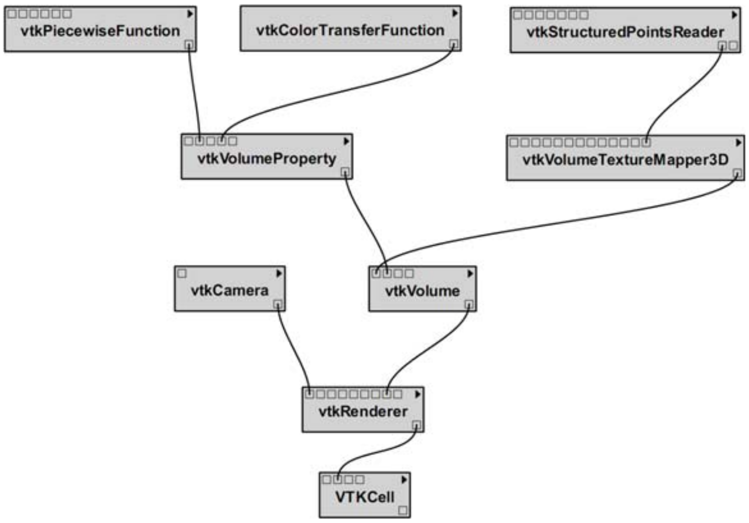


Figure 2.2: A workflow describing a data visualization. Boxes indicate processes and lines indicate data dependencies.

through undefined ports whose values are produced at run time. In a connection $\{(m_i, port_i, m_j, port_j)\}$, the value output on $port_i$ of module m_i is used as input for $port_j$ of module m_j. A set of modules M along with a set of connections C define a partial order relation PO_M on M. This partial order does not contain cycles—the workflow is a directed acyclic graph, or DAG—and defines the execution order of modules.

2.2.2 Scientific Workflow Systems

Although many scientific workflow systems are constructed for a specific application area, the versatility of the workflow concept makes many of them applicable to other domains. We will here give some notable examples: Taverna [Tav], used in bioinformatics, and also in music, meteorology, and medicine; Chimera [FVWZ02], Pegasus [DSS$^+$05] and VDS [VDS] are used for grid computing; VisTrails [Vis] provides support for data exploration and visualization; Kepler [Kep], Swift [Swi], and Triana [Tri] are domain independent workflow systems designed to support multiple domains.

There are also non-scientific workflow systems such as Microsoft Workflow Foundation [Mic], Yahoo! Pipes [Yah] and Apple's Mac OS X Automator [App]. They are commonly used by people without scientific background but are built on the same principles as scientific workflow systems. These tools have the potential to attract a large number of users and generate large workflow collections. This makes them suitable for larger studies of workflow reuse, which can benefit the scientific community.

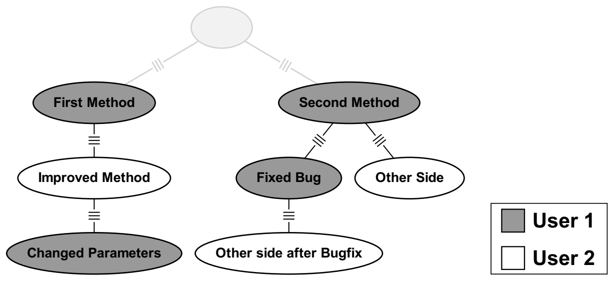


Figure 2.3: An example of collaborative design. Here, two persons have built on each other's workflow specifications, leading to incrementally better workflows.

Yu et al. created a taxonomy of scientific workflow systems for grid computing [YB05a]. This taxonomy shows many important aspects of workflow systems and distinctions between them.

The scenarios described in Figure 2.1 are based on close collaboration, but sometimes several people need to work on a single workflow. Designing the workflow is a task that requires knowledge of the components in the workflow, and different people may have experience with different components. This scenario is described in Section 2.2.3.

The following sections describe the specific challenges that were introduced in this section.

2.2.3 Collaboratively Designing Workflows

In today's scientific community, it is rarely the case that novel scientific discoveries can be made by a single person [FKSS08]. Unfortunately, in many instances of close collaboration, the various domain experts are unable to work in the same location. These types of relationships benefit greatly from the ability to concurrently modify a given workflow description. Here, we explore the benefits of real-time, synchronous collaborative workflow design.

Collaborative Design in Multi-disciplinary Research.

An example of the advantages gained from collaboratively designed workflows can be seen in collaborations between the authors at the University of Utah and researchers at the Center for Coastal Margin Observation and Prediction (CMOP).[1] CMOP scientists, located in Oregon and Washington, often spend a significant amount of time describing the various processing and analysis methods they employ to understand their data. While in many cases e-mail is satisfactory for sharing knowledge with collaborators, in some situations, a more immersive collaborative workspace is required.

When a task relating to a specific researcher's area of expertise is being considered, it is often necessary to synchronize processing workflows to

[1]http://www.stccmop.org

arrive at a desired result. By allowing scientists at the CMOP centers in Oregon to work synchronously with researchers at the University of Utah, the critical task of communication is enriched. Instead of relying on e-mail and telephone conversations to ask important, and often time-consuming, questions, scientists can explore *and fix* each other's processing and parameterization errors in real time. This degree of collaborative design reduces the number and severity of communication-based misunderstandings as well as increasing the level of productivity of everyone involved in the project.

Collaborative design has been explored in other areas such as visualization [WBHW06], but not for workflows in general. Our goal is to be able to collaboratively design a workflow efficiently. This thesis presents a framework that makes real-time collaborative workflow design possible in a workflow system (See Chapter 4). By carefully examining the working process of existing collaborative research projects, we have been able to design a system that not only respects individual working habits, but also strengthens and enhances the interaction among multiple users engaged in collaborative efforts.

2.2.4 Reusing Workflows

Software reuse has always been envisioned as a means to reduce work by building on previous work, and writing reusable code is an important part of software engineering [FF95]. This is equally important for workflows. There are many advantages to reuse [YB05b]. Although workflows have a simple composition model, they are still difficult to assemble, especially since they are used partly by people without programming skills [VVS+08]. Workflow reuse can make workflow creation easier since it is often easier to reuse old workflows than it is to start from scratch. Also, users can learn much from studying working workflows.

Goderis et al. described seven bottlenecks to workflow reuse [GSLG05]. They consist of problems with the current technology that limits the ability to support workflow reuse. The problems can be summarized into five topics: limited service availability; rigidity and inflexibility of workflow models making interoperability between workflow models hard; unclear intellectual property rights on workflows; lack of discovery models; and inability to semantically interpret workflows. These cover different aspects of the workflow life cycle. Workflows are created, executed, distributed, and finally modified and executed by other users [GDE+07b]. But there are many parts of this cycle that need to be improved to make the reuse process more efficient.

Reusability is dependent on the users' ability to share and reuse each other's work by finding and interpreting relevant information. For this to be feasible there needs to be a bigger gain from reusing a workflow than there is in creating one from scratch. The effort to reuse a workflow should be less compared to designing a new one. This effort depends on both previous experience and the complexity of the task. The efficiency of workflow reuse depends on how easy it is to locate and understand them. We would like to be able to locate relevant workflows with minimal effort and understand them with minimal difficulty. This would require that users be willing to share and document their work, and that they trust work produced by others. There should exist tools both for locating workflows and assessing their relevance and validity. Workflow design tools should make it easy to annotate and enrich workflows and should support tools for workflow sharing and collaboration.

There are several examples of projects that addresses workflow reuse. We will describe two of them in short.

myExperiment [GR07] is a web site where workflows can be published and queried. Its main purpose is to bootstrap workflow reuse and to explore the use of social networking in building communities around workflow management. Both the social and technical aspects are neccesary to realize the vision of efficient workflow reuse. The site supports workflows from their own Taverna [Tav] workflow system as well as workflows of other types. Users are encouraged to contribute their workflows and to create communities where workflows can be reused.

Yahoo! Pipes [Yah] is a web site containing workflows that processes, filters, and creates mash-ups (combining information from different sources) of web feeds and other online data sources. Users can both build and execute workflows directly through the web site. The search interface enables queries on module types, parameter values, tags and descriptions. It has a large collection of workflows, but the available module types are few in number, which makes the workflow parameters extra important when considering reuse.

In these projects search capabilities are an essential part of workflow reuse. In a workflow search, users can specify their requirements and browse candidate workflows. A search engine must support two important tasks: querying and displaying the results. While there has been work on the former [SSC09, BEKM06, SKV+07, SVK+08], the latter has been largely overlooked. This thesis proposes methods for displaying summarizations of workflow search results(See Chapter 5). It shows that current approaches to generating workflow search results do not consider structure, and suggests how new approaches can achieve better results.

2.3 Data Provenance

Provenance denotes the history of things, like the ownership history of a painting, or a computational process that led to a specific result. In the context of scientific workflows, *data provenance* is a record of the derivation of a set of results. There are two distinct forms of provenance [CFH+08]: prospective and retrospective.

Prospective provenance captures the specification of the workflow—it corresponds to the *steps that need to be followed* (or a recipe) to generate

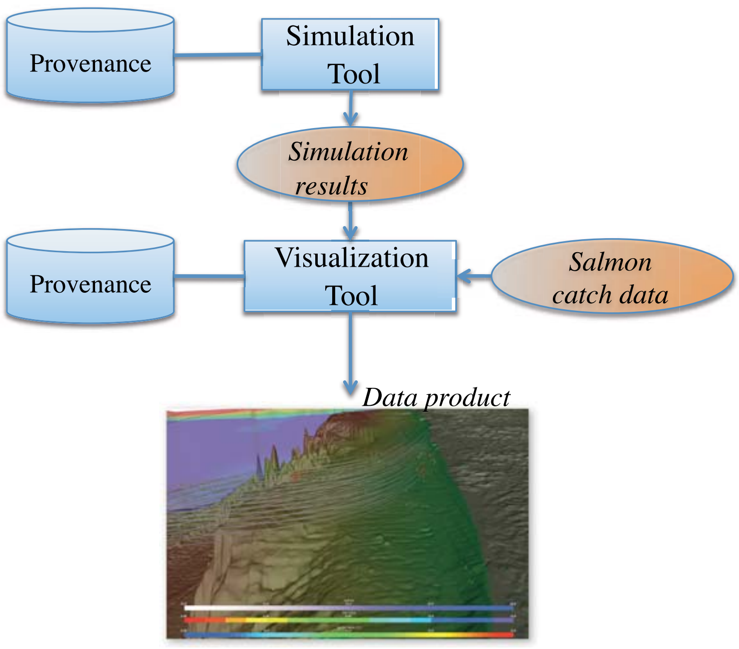


Figure 2.4: The visualization on the bottom shows salmon catch information superimposed with a model of the currents in the mouth of the Columbia River. The simulation was run on a cluster of computers using a grid-enabled workflow system. A workflow-based visualization system was then used to display the simulation together with the salmon catch data.

a data product or class of data products. This recipe will be shared by all data products created by it. Its provenance is usually incomplete, some information is not available until runtime when abstract workflows are instantiated and assigned hardware and data products are given identifiers. *Retrospective provenance* captures the *steps that were executed* as well as information about the execution environment used to derive a specific data product—a detailed log of the execution of the workflow. Ideally, it should contain all information needed to recreate the data product and check its validity.

An important piece of information present in workflow provenance is information about *causality*: the dependency relationships among data products and the processes that generate them [FKSS08]. Causality can be inferred from both prospective and retrospective provenance and captures the sequence of steps which, together with input data and parameters, caused the creation of a data product. Causality consists of different types of dependencies. Data-process dependencies (e.g., the fact that the visualization in Figure 2.4 was derived by a particular workflow run within the visualization tool) are useful for documenting the data generation process, and they can also be used to reproduce or validate the process. For example, it would allow new visualizations to be derived for different input data sets (i.e., different simulation results). Data dependencies are also useful. For example, in the event that the simulation code used to generate simulation results is found to be defective, data products that depend on those results can be invalidated by examining data dependencies.

Although different workflow systems use different data models, storage systems, and query interfaces, they all represent the notion of causality using a directed acyclic graph (DAG). In this graph, vertices are either data products or processes and the edges represent the causal relationships between them.

One of the issues with data provenance is to make it interoperable between workflow systems, such that data can be traced between workflow systems. This would also enable provenance tools and query languages to be shared between workflow systems. Section 2.3.1 discusses this issue.

2.3.1 The Need for Interoperable Provenance

Ideally there would be a single workflow concept that could fit all purposes and be universally accepted. In reality, the formats differ greatly and module libraries are often only available for a single workflow system. Workflows can not easily be translated between workflow systems because of their differing functionalities [GDE+07a]. There is however one aspect that has proven to be important [FKSS08, SPG05]: The data provenance of results generated from workflows. This provenance is generally independent of the workflow used to create them and provenance is important for the validity and reproducibility of data items.

Currently there is no standard way to record the provenance of data, but such standards are in development [MF06]. Even though a standard for workflows may be unobtainable and even undesirable due to different requirements on workflow systems, interoperable data provenance is a crucial component in scientific collaboration. Here is an example illustrating the problem:

> A researcher is using data generated by another research group in a workflow. The data does not give the expected results and it has become necessary to investigate its provenance to check that the data is correct. The data derivation path has been recorded by two separate workflow systems, and their provenance is not compatible. The researcher has two choices, either investigate the provenance data by hand, or gain access to the provenance tools used by the other research group.

In this scenario provenance interoperability would be needed. With a global provenance model, tools can be constructed that can interpret provenance through different sources and be used independently of the workflow system. Provenance interoperability is a topic that has recently started to receive attention in the scientific workflow community [G+07, PRO07]. Most workflow systems support provenance capture, but each adopts its own data and storage models [FKSS08, DF08]. These range from specialized Semantic Web languages (e.g., RDF and OWL) and XML dialects that are stored as files in the file system, to tables stored in relational databases. These systems have also started to support queries over provenance information [Mor08]. Their solutions are closely tied to the data and storage models they adopt and require users to write queries in languages like SQL [BD08] and SPARQL [ZGST08, KDG+08, GH08]. Consequently, determining the *complete* lineage of a data product derived from multiple systems requires that information from these different systems and/or their query interfaces be integrated.

This thesis proposes a provenance integration approach that bridges different provenance models using a data mediator. It shows how this is a scalable and efficient solution in the absence of an accepted provenance standard. The approach is described in Chapter 3.

2.4 Research Strategy

The goal of this thesis is to "contribute to the improvement of scientific collaboration through the use of workflows and provenance". The research strategy of this thesis has been to contribute on topics that require additional work in order for workflows and provenance to reach their full potential. In this section, we describe the research strategy for achieving the contributions in this thesis. Each contribution is described in its own section which contain the following parts: motivation, research question, methodology, and results.

2.4.1 Investigating Provenance Interoperability through Mediation

Chapter 3 describes the problem of provenance interoperability. Data provenance from three different workflow systems are integrated using a database mediation approach. This approach tries to address the issues with the more established data conversion approach. In the study of this case, three specific provenance models are integrated using a global provenance model and a mediator that performs query translation.

Motivation. The motivation for this project was, as part of a bigger collaboration, to advance the interoperability of provenance models, which can promote reusable tools and scientific collaboration.

Research question. What are the advantages and issues with the mediation approach compared to the data conversion approach?

Methodology. Related literature was studied to find the most suitable integration approach. The mediation approach was implemented as a case

study with three different models. The study was then compared with the more commonly used approach: to use direct schema translation and perform queries over a single schema.
Results. Experiences of the method were presented and advantages with the different approaches were discussed.

2.4.2 A Method for Real-Time Collaborative Workflow Design

Chapter 4 describes an implementation of real-time collaborative design mode for a workflow system. It relies on the provenance of the workflow to be able to track changes in real time.
Motivation. Real-time collaborative design was a requested feature that was unexplored at the time and is an important part of the workflow life cycle.
Research question. How can multiple persons work on the same workflow simultaneously?
Methodology. We designed a method for collaborative workflow design that uses workflow design provenance and implemented a prototype as proof-of-concept.
Results. The description of the method and a working prototype.

2.4.3 Presenting Workflow Search Results

The presentation of search results are important when searching any data collection. Certain aspects of workflows makes it necessary to explore new techniques for presenting workflows as search results. In Chapter 5 we describe efficient methods for generating workflow snippets.
Motivation. Available methods for summarizing workflows does not give sufficient information in the summaries. Available literature suggests that methods used for documents is not enough for presenting summarizations of workflows. Instead we propose to use the inherent structure of workflows to summarize workflows.
Research question. How can the workflow structure be used to present search results?
Methodology. We identified suitable requirements on workflow snippets. We developed alternative strategies to generate workflow snippets. These strategies were validated in a user study. Users ranked different strategies and validated the relevant information used to generate the snippets. We explored strategies for presenting sets of workflows.
Results. A set of requirements on workflow snippets. Results of the user study that show the advantages of our methods.

Chapter 3

Using Mediation to Achieve Provenance Interoperability

This chapter describes a mediator-based architecture for integrating provenance information from multiple sources. The architecture contains two key components: a global mediated schema that is general and capable of representing provenance information represented in different models; and a new system-independent query API that is general and able to express complex queries over provenance information from different sources. We also present a case study where we show how this model was applied to integrate provenance from three provenance-enabled systems and discuss the issues involved in this integration process. The work described in this chapter has previously been published as [EKF$^+$09].

3.1 Introduction

Data provenance (as described in Section 2.3) is essential in scientific experiments and is an important part of the scientific process. As described in Section 2.3.1, interoperable provenance is important in collaborations spanning multiple workflow systems. This chapter will describe an approach that makes scalable provenance integration possible. Consider the scenario in Figure 2.4. In order to determine the provenance of the visualization (shown on the bottom), it is necessary to combine the provenance information captured both by the workflow system used to derive the simulation results and by the workflow-based visualization system to derive the image. Without combining this information, it is not possible to answer important questions about the resulting image, such as, for example, the specific parameter values used in the simulation.

This chapter addresses the problem of provenance interoperability in the context of scientific workflow systems. In the Second Provenance Challenge (SPC), several groups collaborated in an exercise to explore interoperability issues among provenance models [PRO07]. Part of the work described here was developed in the context of the SPC.

Although existing provenance models differ in many ways, they all share an essential type of information: the provenance of a given data product consists of a *causality graph* whose nodes correspond to processes and data products, and edges correspond to either data or data-process dependencies. Inspired by previous works on information integration [Wie92], we propose a mediator architecture which uses the causality graph as the basis for its global schema for querying disparate provenance sources (Section 3.2). The process of integrating a provenance source into the mediator consists of the creation of a wrapper that populates the global (mediated) schema with information extracted from the source. As part of the mediator, we provide a query engine and an API that supports transparent access to multiple provenance sources. We evaluate the efficiency of this approach by applying it to integrate provenance information from three systems (Section 3.3). We discuss our experiences in implementing the system and its relationship to recent efforts to develop a standard provenance model.

3.2 A Mediation Approach for Integrating Provenance

Information mediators have been proposed as a means to integrate information from disparate sources. A mediator selects, restructures, and merges information from multiple sources and exports a global, integrated view of information in these sources [Wie92]. In essence, it abstracts and transforms the retrieved data into a common representation and semantics. An information mediator consists of three key components (see Figure 3.1): a global schema that is exposed to the users of the system; a query rewriting mechanism that translates user queries over the global schema into queries over the individual data sources; and wrappers that access data in the sources and transform them into the model of the mediator.

In what follows, we describe the mediator architecture we developed for integrating provenance information derived by scientific workflow systems. In Section 3.2.1, we present the global schema used and in Section 3.2.2 we discuss the query API supported by our mediator. Details about the wrappers are given later, in Section 3.3, where we describe a case study which shows that the data model and query API can be efficiently used to support queries over (real) provenance data derived by different systems.

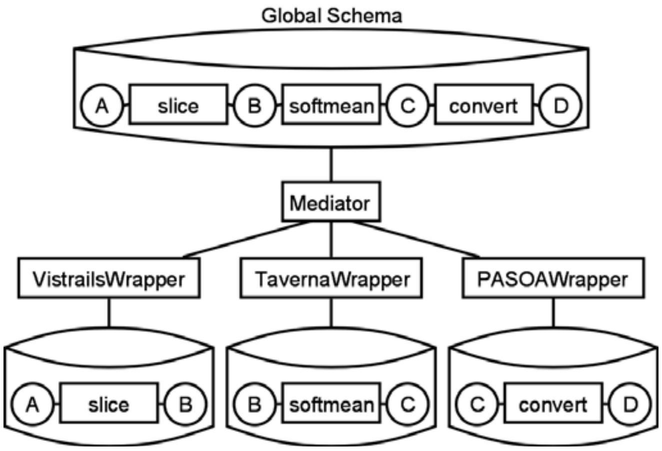


Figure 3.1: Mediator architecture used to integrate three provenance models. Queries over the global schema are translated by the wrappers into queries over the provenance sources, which are then executed and their results returned to the mediator. In this example, pieces of a complex workflow (*slice*, *softmean* and *convert*) were executed by the workflow systems. *A*, *B*, *C*, and *D* are data items.

3.2.1 A Data Model for Scientific Workflow Provenance

The provenance causality graph, as introduced in Section 2.3, forms the basis for the global schema used in our mediator architecture. A central component of our mediator is a general provenance model, the *Scientific Workflow Provenance Data Model (SWPDM).* The model captures entities and relationships that are relevant to *both* prospective and retrospective provenance, i.e., the definition and execution of workflows, and data products they derive and consume. As a result, besides queries over provenance, our model also supports direct queries over workflow specifications. As we discuss later, this is an important distinction between SWPDM and the Open Provenance Model [MFF$^+$07].

The entities and relationships of the SWPDM are depicted in Figure 3.2. At the core of the model is the *operation* entity, which is a concrete or abstract data transformation, represented in three different layers in the model: *procedure*, which specifies the type of an operation; *module*, which represents an operation that is part of an abstract process composition; and *execution*, which represents the concrete execution of an operation.

A *data item* represents a data product that was used or created by a workflow. A *procedure* represents an abstract operation that uses and produces data items. A *procedure declaration* is used to model procedures together with a list of supported *input* and *output ports*, which have types (e.g., integer, real). It describes the signature of a module. A port is a slot from/to which a procedure can consume/produce a data item. A *workflow*

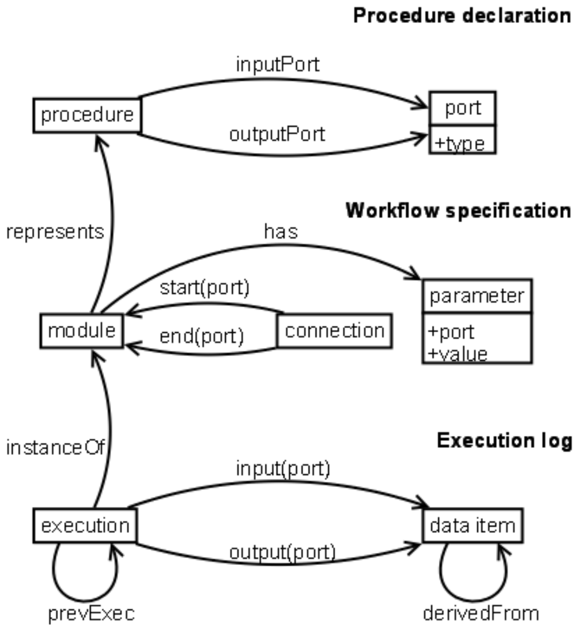


Figure 3.2: Overview of the Scientific Workflow Provenance Data Model. Boxes represents entity types and arrows represents relationships between entities. Relationships can have attributes, shown after the relationship name. The *procedure declaration* specifies a list of typed input/output ports for each procedure—the signature of the procedure. The *workflow specification* contains the modules, connections and parameters that makes up the workflow. The *execution log* contains a record of executions of processes and used data items.

specification consists of a graph that defines how the different procedures that compose a workflow are orchestrated, i.e., a set of *modules* that represent procedures and *connections* that represent the flow of data between modules. *Parameters* model predefined input data on specific module ports. The *execution log* consists of concrete *executions* of procedures and the data items used in the process.

Two points are noteworthy in our choice of entities. A workflow is assigned input data before execution, but besides these inputs, a module may also have parameters that serve, for example, to set the state of the module (e.g., the scaling factor for an image). From a provenance perspective parameters are simply data items, but by using a finer grained division into parameters we can support more expressive queries. Furthermore, by modeling workflow *connections* as separate from the data items we are able to query the structure of a workflow directly. These connections can then be used to answer queries like "which data items passed through this connection".

Function	Description
outputOf(data)	get execution that created *data*
inputOf(data)	get execution that used *data*
output(execution)	get data created by *execution*
input(execution)	get data used by *execution*
execOf(execution)	get the module representing *execution*
getExec(module)	get the executions of *module*
represents(module)	get the process that *module* represents
hasModule(process)	get module that represents *process*
derivedFrom(data)	get data products used to create *data*
derivedTo(data)	get data products derived from *data*
prevExec(execution)	get execution that triggered *execution*
nextExec(execution)	get executions triggered by *execution*
upstream(x)	transitive closure operation, where x is a module, execution or data
downstream(x)	transitive closure operation,where x is a module, execution or data

Table 3.1: List of API functions.

Another key component of workflow provenance is *user-defined information*. This includes documentation that cannot be automatically captured but records important decisions and notes. This information is often captured as annotations, which are supported by most workflow systems. In our model, an *annotation* is a property of entities in the model. Annotations can add descriptions to entities that can later be used for querying. Examples are execution times, hardware/software information, workflow creators, descriptions, labels, etc. In the model we assume that any entity can be annotated.

3.2.2 Querying SWPDM

We have designed a new query API that operates on the entities and relationships defined in the SWPDM. This API provides basic functions that can serve as the basis for implementing a high-level provenance query language. In order to integrate a provenance source into the mediator, one must provide system-specific bindings for the API functions. Figure 3.3 illustrates the bindings of the API function $getExecutedModules(wf\_exec)$ for three distinct provenance models. As we describe in Section 3.3.2, each binding uses the data model and query language supported by the underlying system.

Note that a given workflow system may not capture all the information represented in our model (see Figure 3.2). In fact, the systems we used in our case study only partially cover this model (see Section 3.3). Thus, in designing API bindings for the different systems, the goal is to extract (and map) as much information as possible from each source.

The core API functions are summarized in table 3.1.[1] Since the API operates on a graph-based model, a key function it provides is graph traversal. The graph-traversal functions are of the form *getBFromA*, which traverse the graph from A to B. For example, *getExecutionFromModule* traverses the graph from a module to its executions, i.e., it returns the execution logs for a given module.

Additional functions are provided to represent common provenance operations which have to do with having both data- and process-centric views on provenance. For example, *getParentDataItem* returns the data items used to create a specific data item. Such parent/child functions also exist for modules and executions.

Note that the API contains redundant functions, e.g., *getParentExecution* can also be achieved by combining *getExecutionFromOutData* and *getInDataFromExecution*. If data items are not recorded by the given provenance system, the binding for *getParentExecution* might use another path to find the previous execution. Although these redundant functions are needed for some provenance systems, they can make query construction ambiguous. It is up to the wrapper designer to implement these based on the capabilities of the underlying system and its data model.

Provenance queries often require transitive closure operations that traverse the provenance graph and tracing dependencies forward or backward in time. Our API supports transitive closure queries in both directions: *upstream*, which traces provenance backwards (e.g., what derived a given data item); and *downstream*, which traces provenance forward (e.g., what depends on a given data item).

The transitive functions are represented as $y = upstream(x)$ where x is an entity and y represents all its dependencies. There is also a corresponding downstream function. Since these queries can be expensive to evaluate, it is useful to have additional attributes that can prune the scope of the search. A *depth restriction* specifies the maximum depth to explore, and a *scope restriction* specifies entities that should be ignored in the traversal. These restrictions are captured by the function $y = upstream(x, depth, scope)$, and the corresponding downstream function.

There are additional operations; *getAll* returns all entities of a specific type and is used to create local caches of the provenance store. Two operations handle annotations: *getAllAnnotated* returns entities containing a specific annotation and is used for queries on annotations; *getAnnotation* returns all annotations of an entity.

[1]The complete API, and the bindings to Taverna, PASOA and VisTrails, are available for download at:
http://twiki.ipaw.info/pub/Challenge/VisTrails2/api.zip

VisTrails (XPath)

```
def getExecutedModules(self, wf_exec):
    newdataitems = []
    q = '//exec[@id="' + wf_exec.pid.key + '"]/@moduleId'
    dataitems = self.logcontext.xpathEval(q)
```

Pasoa (XPath)

```
def getExecutedModules(self, wf_exec):
    q = "//ps:relationshipPAssertion[ps:localPAssertionId='" +
               wf_exec.pid.key + "']/ps:relation"
    dataitems = self.context.xpathEval(q)
```

Taverna (SPARQL)

```
def getExecutedModules(self, wf_exec):
    "  "
    q = '''
    SELECT ?mi
    FROM <''' + self.path + '''>
    WHERE
      { <''' + wf_exec.pid.key + '''>
         <http://www.mygrid.org.uk/provenance#runsProcess> ?mi }
    '''
    return self.processQueryAsList(q, pModuleInstance)
```

Figure 3.3: Implementations of the *getExecutedModules* function for different provenance systems.

3.2.3 Discussion and Related Work

Mediator Architecture. There are a few different approaches to mediation [Hal03]. In this chapter, we explore the virtual approach, where information is retrieved from the sources when queries are issued by the end user. Another approach is to materialize the information from all sources in a warehouse. The trade-offs with each approach are well-known. Whereas warehousing leads to more efficient queries, data can become stale,and the storage requirements can be prohibitive. Nonetheless, our architecture can be used to support a warehousing solution: once the wrappers are constructed, queries can be issued that retrieve all available data from the individual repositories to be stored in a warehouse.
Other Approaches to Provenance Interoperability. Thirteen teams participated in the Second Provenace Challenge, whose goal was to establish provenance interoperability among different workflow systems. Most solutions mapped provenance data from one system onto the model of another. Although all teams reported success, they also reported that the mapping process was tedious and time consuming. To create a general solution, such approaches would require n^2 mappings, where n is the number of systems

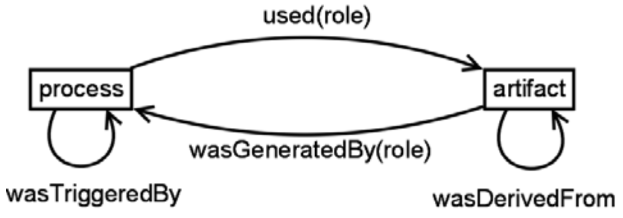


Figure 3.4: The basic concepts in the OPM. It maps directly to the execution log of our SWPDM. The entity types and relationships have different names but represent the same concepts. Here, data items are called *artifacts* and ports are called *roles*.

being integrated. In addition, they require that all data from the different systems be materialized, which may not be practical. In contrast, by adopting a mediator-based approach, only n mappings are required—one mapping between each system and the global schema. And as discussed above, both virtual and materialized approaches are supported by the mediator architecture.

Provenance interoperability is a new research area, but the data warehouse approach has already been studied as part of the Provenance Challenge [PRO07] by teams using different workflow systems. Our study is also part of this project and uses the same queries and data models. However, none of the other teams attempts the mediation approach, possibly because its implementation is more time consuming. However, when considering performance and scalability, the mediation approach seems more attractive.
The Open Provenance Model. One of the outcomes of the Second Provenance Challenge was the realization that it is indeed possible to integrate provenance information from multiple systems, and that there is substantial agreement on a core representation of provenance [PRO07]. Armed with a better understanding of the different models, their query capabilities, and how they can interoperate, Moreau et al. [MFF+07] proposed a standard model for provenance: the Open Provenance Model (OPM). The OPM defines a core set of rules that identify valid inferences that can be made on provenance graphs. Important goals shared by the OPM and SWPDM include: simplify the exchange of provenance information, and allow developers to build and share tools that operate on common models. However, unlike the SWPDM, OPM supports the definition of provenance for any "thing", whether produced by computer systems or not. In this sense, OPM is more general than SWPDM. However, by focusing on workflows and modeling workflow representations, SWPDM allows a richer set of queries that correlate provenance of data products and the specification of the workflows that derived them.

We should note that many of the concepts model by OPM can also be modeled in SWPDM. Most of the OPM concepts have a direct translation to SWPDM. Figure 3.4 shows the OPM representation of the relationships between processes and data items. This representation can be mapped directly to the execution log of SWPDM, which contains the provenance graph. The OPM also contains a number of inferred relationships, namely transitive versions of the basic relationships. We support these by using transitive functions in the API (upstream and downstream, see Section 3.2.2). The OPM has an optional part component to represent time. SWPDM supports time as an annotation, which means we can support any number of representations of time including the one in the OPM. In the OPM there is also a notion of an *agent* that is responsible for executions. These can also be modeled as annotations. Finally the OPM enables multi-level provenance, i.e., provenance at different *granularities*, using the notion of *accounts*. Each process and artifact is assigned one or more accounts. Each query uses one or more accounts to access different levels of the provenance. This is something that is not present in the SWPDM. Instead, we assume that the underlying query engine already has one specific granularity predetermined, either by checking the user's level of preferred access or accessing a provenance store of specific granularity. It is still an open question how this will work in practice. The accounts in the OPM have not yet been tested. The OPM was not available during the project. It would be possible to implement a wrapper for the OPM or even to use it as the global model. But the OPM is a lightweight model and specific implementations of the OPM can differ. OPM uses extensions called *profiles* that extend its functionality. A few suggested profiles include binding to Dublin Core [WKLW98] concepts, time annotations, and support for data collections. Once fully developed, there may be solutions to the definition of identifiers and namespaces. But it is not clear whether all implementations of OPM would adopt the same profiles and a more thorough investigation is needed.

3.3 Case Study: Integrating Provenance from Three Systems

We implemented the mediator architecture described in Section 3.2 as well as bindings for the query API using three distinct provenance models: VisTrails [Vis], PASOA [GMM05], and Taverna [Tav]. Figure 3.1 shows a high-level overview of our mediator.

In order to assess the efficiency of our approach, we used the workflow and query workload defined for the Provenance Challenge [PRO07]. The workflow entails the processing of functional magnetic resonance images, and the workload consists of typical provenance queries, for example: What was the process used to derive an image? Which data sets contributed to the derivation of a given image? For a detailed description of the workflow and queries, see [PRO07]. Before we describe our implementation and experiences, we give a brief overview of the provenance models used in this case study.

3.3.1 Provenance Models

VisTrails is a scientific workflow system developed at the University of Utah. A new concept introduced with VisTrails is the notion of *provenance of workflow evolution* [FSC+06]. In contrast to previous workflow systems, which maintain provenance only for derived data products, VisTrails treats the workflows (or pipelines) as first-class data items and keeps their provenance. The availability of this additional information enables a series of operations that simplify exploratory processes and foster reflective reasoning, for example: scientists can easily navigate through the space of workflows created for a given exploration task; visually compare workflows and their results; and explore large parameter ranges. VisTrails captures both prospective and retrospective provenance, which are stored uniformly either as XML files or in a relational database.

Taverna is a workflow system used in the myGrid project, whose goal is to leverage semantic web technologies and ontologies available for bioinformatics to simplify data analysis processes in this domain. Prospective provenance is stored as Scufl specifications (an XML dialect) and retrospective provenance is stored as RDF triples in a MySQL database. Taverna assigns globally unique LSID [ZGS06] identifiers to each data product.

PASOA (Provenance Aware Service Oriented Architecture) relies on individual services to record their own provenance. The system does not model the notion of a workflow, instead, it captures assertions produced by services that reflect the relationships between services and data. The complete provenance of a task or data product must be inferred by combining these assertions and following the relationships they represent. PReServ, an implementation of PASOA, supports multiple back-end storage systems, including files and relational database, and queries over provenance can be pose using its Java-based query API or XQuery.

3.3.2 Building the Mediator

Our model was developed based on a study of the three models above. Each model covers only part of the mediated model. For both Taverna and PASOA, only the execution log was available: The workflow specifications were not provided.

VisTrails stores both workflow specification and the execution log. It uses a normalized provenance model where each execution record points to the workflow specification where it came from. The system does not explicitly identify data items produced in intermediate steps of workflow execution.

Implementation. We implemented the mediator-based architecture in Python. The different components are shown in Figure 3.5. *PQObject* represents a concept in the global schema; *PQueryFactory* the mediator;

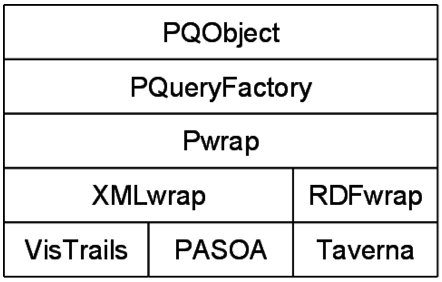


Figure 3.5: The different layers in the implementation of the query API. Queries are processed starting from a known entity (*PQObject*) and traversed by using relationship edges in the mediator (*PQueryFactory*). The mediator executes the query using the wrapper interface (*Pwrap*), which in turn executes the query using a specific wrapper for each data source.

and *XMLwrap* (for XML data using XPath) and *RDFwrap* (for RDF data using SPARQL) are abstract wrappers. The concrete wrappers are in the bottom layer: for PASOA and VisTrails, wrappers were built by extending XMLWrap; and for Taverna using RDFWrap. These wrappers implement API functions defined in Section 3.2.2 using the query interfaces provided by each system. Due to space limitations, we omit the details of the API bindings. The source code for the mediator and bindings is available at http://twiki.ipaw.info/pub/Challenge/VisTrails2/api.zip.

Using and Binding the API. Here we show some examples of how the API functions can be used to construct complex queries. Since each function applies to an entity, first it is necessary to obtain a handle for the entity instance of interest. For example, to access the handle for a port, the node corresponding to that port needs to be extracted from the global schema: $m = pqf.getNode(pModule, moduleid, store1.ns)$. This method accesses the components of an instance of the PQueryFactory, and it requires the specification of the entity type (*pModule*), entity identifier (unique identifier), and location of the entity (a specific provenance store). Once the handle has been retrieved, the provenance graph is traversed by invoking an API function (e.g., to get all executions of a module e: $d = e.getDataItemFromExecution()$).

There are some issues that need to be considered during query construction. First, there are different ways to represent a workflow. For example: modules can contain scripts whose parameters are not properly exposed in the module signature; and parameters can be modeled as input ports. Thus, the actual implementation of a query depends on the chosen representations and semantics implemented within the wrapper. Another issue concerns the specification of data items and inputs. Some systems record the concrete data item used as input while others represents data items by names that are stored as parameters to modules that use the data item. This must also be resolved during the wrapper design, by modeling data items, inputs and parameters consistently across distinct provenance stores.

Consider, for example, provenance challenge query 6 which asks for "the process that led to the image Atlas X Graphic", i.e., the provenance graph that led to this specific image. In VisTrails, the image is identified by the string *atlas-x.gif* that is specified as a parameter to a FileSink module in the workflow specification. In contrast, Taverna uses a string representing a port name *convert1_out_AtlasXGraphic*—here, data items are handled internally and are not saved to disk. PASOA uses the string *atlas-x.gif* that is passed between the two modules. This means that the starting handle obtained for the different systems will be of different types. In VisTrails it is a parameter to a module, in Taverna it is an Output port for a module, and in PASOA it is a data item. Thus, once the handle for "Atlas X Graphic" is obtained, different methods need to be used in order to compute the required `upstream`. For VisTrails the FileSink module that contains the file name is located, then the executions of that module are found and finally the upstream of those executions are returned. For Taverna, the executions associated with the output port are located and used to run the upstream. For PASOA the executions that produced the data item are located and used to run the upstream.

3.3.3 Complex queries

Complex queries are queries that contains joins and cannot be expressed by simply traversing the graph. Provenance challenge query 6

> "Find all output averaged images of softmean (average) procedures, where the warped images taken as input were align_warped using a twelfth order nonlinear 1365 parameter model"

is a complex query in the sense that it requires joins. It is not sufficient to traverse the provenance graph, the restrictions of the entities need to be joined to produce the correct result, e.g., to find the correct *align_warp* execution, locate the executions that are of type *align_warp* and that have a specific annotation. These two restrictions need to be joined to find the correct executions. This requires the queries to be executed in in multiple steps and suggests using a higher-level declarative query language to express them.

Using our API, we plan to implement a high level query language that will be capable of expressing these kind of complex queries. Here we describe how a high-level query language could be used to answer query 6.

```
data: aw.type = 'execution' and aw.procedure = 'align\_warp' and
      aw.parameter('argument') = '-m 12' and sm in upstream(aw) and
      sm.procedure = 'softmean' and data = sm.outData
```

Here, “data:” means return ‘data’ where...”. There are several implicit joins in this query together with a transitive operation. This notation expresses relationships between objects. It uses the variables *aw* (the execution of *align_warp*), *sw* (the execution of *softmean*) and *data* that is the data items produced by *sm*. There is also an annotation restriction on *aw* and an upstream restriction stating that *sm* is in the upstream of *aw*. A query language with these properties will be able to express a rich set of natural queries while being efficient to process.

3.4 Implementation

The system was implemented using the Python language. Currently, wrappers have been implemented for XML [XML08] using XPath [XPa08] and RDF [RDF08] using SPARQL [SPA08]. Figure 3.5 shows the implementation layers from the model (top) to the data files (bottom). It is a mediator model where *PQObject* represents a concept in the global schema; *PQueryFactory* the mediator; and *XMLwrap* and RDFwrap represents generalized wrappers. On the bottom are the concrete wrappers where VisTrails and PASOA are implemented using XMLwrap and Taverna using RDFwrap.

PQObject represents an entity in our model. It can be used to call API functions in PQueryFactory to traverse the model as a graph. PQueryFactory contains the wrappers and forwards queries to the correct wrapper by checking the entity namespace.

All wrappers inherit from Pwrap. It contains functions implementing bridging between sources (e.g. data item x in source A is dependent on data item y in source B, or execution x in source A received a data item from execution y in source B). It also provides default upstream/downstream functions with source bridging support.

There are currently 2 types of wrappers: XMLwrap implements loading of an XML data file and provides access through XPath. RDFwrap implements access to an RDF server using SPARQL. Other data types i.e. relational DB can be implemented in a similar way.

The wrappers for VisTrails and PASOA have been implemented using XMLwrap while the wrapper for Taverna has been implemented using RDFwrap. Taverna uses the XML/RDF format for specifying provenance, which makes it possible to process it using both XPath and SPARQL. We have chosen to use SPARQL because it is native to the data format and we wanted to have another type of wrapper. We also experimented with implementing it using XMLwrap and found that it was possible but more complex to implement since all the RDF constructs had to be expressed using XPath queries.

3.4.1 Experiences

We will now discuss our findings during the case study as well as issues encountered while implementing the mediator and how they have been addressed.

Mismatches between models. We found only a few mismatches between the models we considered. This indicates that it is possible to create a general provenance model that is efficient. One mismatch was due to the labeling of modules. In Taverna, modules are assigned user-defined names whereas VisTrails uses the name of the module type. This can confuse users, for example, a VisTrails user looking at a Taverna workflow may falsely assume that the label of a module represents its type. In our implementation, this was resolved by creating the concept *label* that for Taverna mapped to the module label and for VisTrails mapped to the module type. The label provides a description of the module. For example, the label of a VisTrails module will be the same as the name of its type.

Implementing wrappers. Constructing wrappers is time-consuming. A wrapper needs to provide a wide range of access methods so that it can efficiently support general queries. This is in contrast to $n * n$ mapping approaches (see Section 3.2.3), where specifying an individual mapping between two models can be a lot simpler. The down side for the latter is that too many mappings may be required for a more comprehensive integration system.

Transitive closure. The implementation of transitive closure (i.e., the upstream/downstream functions) was problematic. XPath and SPARQL does not support transitive closure operations natively. Possible workarounds are to use recursive functions for XQuery [XQu08] or add inference rules to the SPARQL engine. But in both solutions, queries are expensive to evaluate. To overcome this problem, we have used two other methods: (1) wrappers implement the closure as recursive calls of single-step operations—this method is system-independent but not very efficient; (2) cache the transitive relations on the mediator and perform the closure computation on the client side—this method is fast, but not scalable, since it requires loading the transitive data from the data stores. Support for transitive closure computation is a topic that has been well-studied in deductive databases (see e.g., [IR88]) and we plan to experiment with alternative and more efficient approaches.

We note that the optimizations for limiting the scope and depth of transitive closures were useful for some of the provenance challenge queries. Scope means restricting traversal of nodes that are known to be of no interest to the user. Depth means restricting the number of steps the query should traverse. For example, scope restrictions were used to discard everything before *softmean* in Query 2; and for Query 3, the *stages* can be interpreted as meaning depth, in which case a depth restriction is useful.

Provenance interoperability and data identifiers. A problem that needs to be addressed to reconstruct the provenance of a data item across multiple provenance stores is how to identify the data items used, e.g., how to assign a string to each data item that uniquely identifies it. If a data item is the product of a sequence of workflows and the connections among the workflows are represented only by the data items they exchange, then versioning of the files is required to correctly connect the provenance records. Furthermore, identifiers used in the different models may be of different types and use different resolution methods. As a result, one can not assume any specific format or that the identifiers will be unique. Our solution was to append the type of provenance model to the identifier and treat the identifiers themselves as strings. Essentially, our mediator creates its own namespace. It would be preferable to use the namespace already present in the source data but since they were of different types or not present at all this was not possible. Another potential solution for this problem would be the adoption of a common identifier type for provenance entities, such as the LSIDs as used in the Taverna model. This is also an open question in the OPM.

3.5 Conclusions

We addressed the problem of provenance interoperability as data integration and proposed a mediator-based architecture for integrating provenance from multiple sources. We have identified a core set of concepts and access functions present in different provenance systems, and used them to develop a general provenance model and query API. To validate our approach, we implemented the mediator along with wrappers for three provenance models. We used the system to evaluate a series of queries over provenance data derived by the different systems. The preliminary results of our case study indicate that this approach can efficiently integrate data provenance.

Chapter 4

Using Provenance to Support Real-Time Collaborative Design of Workflows

This chapter presents a method for real-time collaborative workflow design. The synchronization of workflow changes, which we call workflow evolution provenance, is automatic, immediate, and unobtrusive, allowing users to see collaborators' changes as they are made. This enables an efficient method of collaboration. We present an interface and algorithm for the synchronization that is based on the common scenarios presented in Section 2.2.3.

The work described in this chapter has been published as [EKA$^+$08].

4.1 Introduction

In collaborative scenarios, users often exchange workflow specifications via e-mail. This process can be slow and tedious and is not practical for situations where results need to be obtained quickly. In some cases, it may be possible to divide the work in such a way that collaborators can work independently and then combine their work for a final result. However, this assumes that a modular design is possible; in reality, workflows are often created through trial and error, with many inter-dependencies.

To support the collaborative design of workflows, we propose a mechanism that allows collaborators to simultaneously work on a task and see each other's changes in real time. With a group of users who are working on the same task, the changes made by each user are automatically propagated to the rest of the group. Note that we *do not automatically merge* changes like

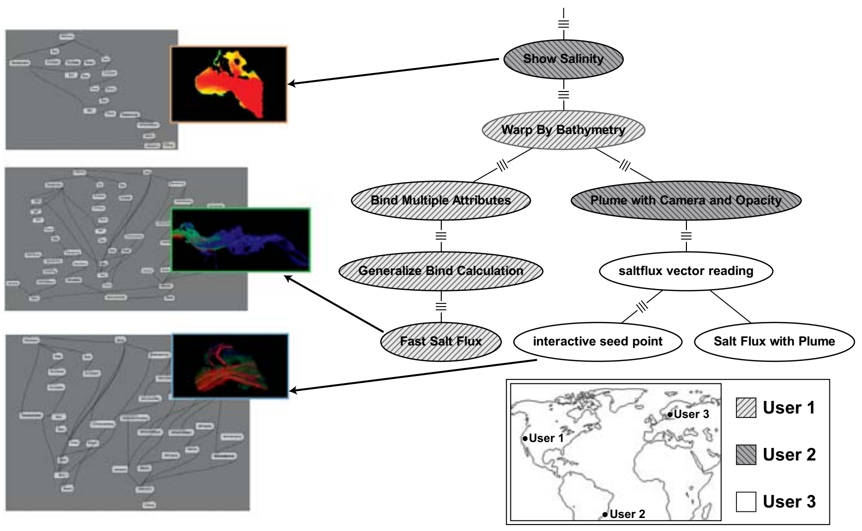


Figure 4.1: A version tree containing a series of workflows that derive visualizations of the Columbia River Estuary. The visualizations have been created by collaborating users. Versions created by different users are represented using different colors.

version control systems. Rather, we display each change as a new branch of exploration and allow the user to switch between branches regardless of who created them. Using workflow evolution provenance, for example, the change-based representation for a collection of workflows [FSC+06], we can visually display a tree containing all contributions. This lets collaborators share and receive updates in real time, while at the same time giving them the option to selectively ignore updates they do not care about. This chapter describes an architecture that supports this functionality. We present a new algorithm for synchronization and discuss how it can be used in practice.

4.2 Architecture

In order to support real-time collaborative design workflows, we need a provenance architecture that supports a collection of versioned workflows and a centralized provenance repository that all collaborators can access. We require a versioning system because each user needs to know how their collaborators' work relates to their own. More importantly, we need to protect the users' work; we should not blindly erase or update their own changes. A centralized repository is needed to manage all the workflows and to provide the means for notifying collaborators when changes occur. The combination of these two methods not only allows users to efficiently share collections of workflows, but also enables them to see the entire history of the workflow specifications as they develop in real time, regardless of how many users collaborate on the project.

Workflow Evolution Provenance. Because we expect to encounter a large number of changes to a workflow specification, especially in a collaborative environment, it can be inefficient to store specifications for all different versions of the workflows. The change-based provenance model [FSC$^+$06] provides a concise representation for workflow evolution history. This model captures the changes applied to a series of workflows, akin to a database transaction log. As a user modifies a workflow (e.g., by adding a module, changing a parameter or deleting a connection), the provenance mechanism transparently records each change action. We can then reconstruct any workflow specification by replaying the sequence of captured changes from an empty specification to the desired version.

The change-based model not only captures changes as a workflow evolves, but it also presents external changes to collaborators in a meaningful way. An important feature of this representation is that it can be visualized as a *version tree*, where each node corresponds to a workflow specification and each edge corresponds to the sequence of changes that transforms the parent specification into the child. Because the version tree captures *all* changes, users have great flexibility in exploring different alternatives without worrying about losing the ability to go back to a specific version. They can perform arbitrary undos and redos—any workflow version is easily recalled by selecting the corresponding node in the version tree. Additionally, users can easily see how their collaborators have taken different approaches to solving related problems and how others' techniques relate to their own ideas. As discussed below, we leverage this layout to inform users of changes without forcing them to immediately consider or integrate those changes.

Centralized Repository. We use a relational database management system (RDBMS) for our centralized repository in order to efficiently capture and broadcast workflow changes. We chose to use a RDBMS because these systems provide secure access protocols, support concurrent transactions from multiple users, and include trigger mechanisms for alerting users when the database is updated. These features are essential to ensure data consistency and to support real-time updates in our collaborative infrastructure. Other kinds of database systems that support these features could also be used in our infrastructure.

To use an RDBMS for our repository, we need to map the necessary provenance information to a relational schema. Because we use the change-based representation, a collection of related workflows is stored as a tree. This tree contains metadata and an ordered set of actions that correspond to user modifications to workflows. Each action, in turn, consists of a sequence of atomic operations. For example, a paste *action* that adds a set of modules and connections to an existing workflow contains a sequence of *operations*: `add module`, `add connection`, *etc.*. An operation, besides its data payload (e.g., module specification, connection specification, parameter value), includes metadata (e.g., the user who performed the action and annotations). Each of these entities (actions, operations, payloads) is stored in its own table, permitting a normalized (redundancy-free) representation. In addition to storing the change-based representation of workflow evolution, the schema also supports explicit workflow specifications and workflow execution information. Execution information can be important when users are unfamiliar with the collection of workflows and wish to know which workflows are routinely used and which workflows were successfully executed.

4.3 Synchronized Design

Our main contribution is a new method for automatically capturing workflow changes performed by multiple users and alerting them about these changes immediately and unobtrusively. This allows users in different geographically distributed locations to collaboratively design and refine workflows, as in the scenario illustrated in Figure 4.1. We accomplish this by committing the local changes (performed by each individual user) to a centralized repository, sending the changes out from the repository to each collaborator, and adding the changes to each collaborator's local version tree. Note that we are not merging workflow specifications but synchronizing workflow evolution provenance. Each collaborator can continue their work and they need not even view the new changes. Before describing the implementation of our prototype, we describe the algorithm for synchronizing the version tree.

4.3.1 Algorithm

There are two key requirements for our algorithm. First, we need a way to save data from a local version tree to the centralized repository. Second, we need a way to load data from that repository to update the collaborators' local version trees. Below, we describe the mechanisms we developed to satisfy these requirements.

Recall that the version tree is induced by a set of actions A. Each action $a \in A$ has a unique identifier derived by the function $id : A \to \mathbb{N}$, where id assigns the smallest unassigned integer to a new action. This function is trivially monotonic: given $a_1, a_2 \in A$,

$$id(a_1) < id(a_2) \iff a_1 \text{ was added before } a_2$$

We will leverage this property to easily determine what has changed in a given version tree. Specifically, let

$$N(A) = \max_{a_i \in V} id(a_i)$$

be the largest action id in a set of actions A. Then, for two sets of actions,

$A_1 \subseteq A_2$, the set of new actions, ΔA, is

$$\Delta A = \{a \in A_2 \mid N(A_1) < id(a) \leq N(A_2)\}$$

This means that we can efficiently determine which actions are required for a user to update his version tree. If a user has copied all of the actions in the database up to id N_D, then we only need to copy actions a_i with $id(a_i) > N_D$ from the database. Conversely, if a user has already saved all actions up to N_L to the database, only actions a_i with $id(a_i) > N_L$ need to be sent to the database. Figure 4.2 shows a simple example of the steps of the algorithm.

Relabeling. Determining the set of new actions is easy when one of the two sets being compared is a superset of the other. However, when multiple users are collaborating, we might not be in this situation. Consider the scenario shown in Figure 4.3, where user A and user B made changes at the same time. Both clients will try to simultaneously save their actions to the database before being notified of the other's changes. In each of their local version trees, they both have actions with id 7, but these actions are not the same. Assuming A's request gets to the repository first, her action will be given id 7 while B's action will become id 8. Thus, after pushing out the other's updates, A and B will have the same tree except that the ids of the nodes may differ.

Since an update of the ids in the local version tree might interfere with a user's current work, we choose to maintain a set of local ids that can be mapped to the global repository ids. Specifically, we maintain a bijective map

$$M : id_{global} \leftrightarrow id_{local}$$

Let M_{local} denote the reverse mapping from global to local and M_{global} denote the forward mapping from local to global. All user operations will be accomplished using the local ids, but whenever we need to save to the centralized repository, we translate everything to the global set of ids. Figure 4.3 shows an example of this relabeling.

Beyond Actions. As described earlier, an action contains metadata and a set of atomic operations. The metadata and the atomic operations, in turn, contain their own ids and may also include references to other entities. Thus, the relabeling of an action needs to update these references as well. For example, each action stores both its own id (*action.id*) and its parent id (*action.prev_id*). If we update the id of the action referenced by *action.prev_id*, we also need to update the *prev_id* field. The same is true for child objects. Suppose the connection in an `add connection` operation references the two modules it connects by id. If we remap the id of one or both of those modules in an `add module` operation, we need to update the ids in the `add connection` operation as well. This requires an ordering that respects the properties being updated; we impose an explicit order on modules and connections so that all modules are relabeled before connections to ensure all references are updated.

Algorithm 1: Incremental Load Algorithm
Input: The local version tree V, id_V (the id function for V), the global-to-local id map M, and the centralized repository D.
Output: None. It updates both V and M in place.

```
LOAD($V$, $id_V$, $M$, $D$)
(1)   $max\_id \leftarrow$ Query $V$ for the maximum id
(2)   $A \leftarrow$ Query $D$ for all actions with id $> max\_id$
(3)   foreach a in A:
(4)       Create $a'$, a local copy of $a$
(5)       $a'.id \leftarrow id_V(a)$
(6)       $a'.prev\_id \leftarrow M_{local}(a.prev\_id)$
(7)       Add pair $(a.id, a'.id)$ to $M$
(8)       Add $a'$ to $V$
```

Algorithm 2: Incremental Save Algorithm
Input: The local version tree V, id_D (the id function for D), the global-to-local id map M, and the centralized repository D.
Output: None. It updates both V and M in place.

```
STORE($V$, $id_D$, $M$, $D$)
(1)   $max\_id \leftarrow$ Query $D$ for the maximum id
(2)   $A \leftarrow$ Query $V$ for all actions with id $> max\_id$
(3)   foreach a in A:
(4)       Create $a'$, a global copy of $a$
(5)       $a'.id \leftarrow id_D(a)$
(6)       $a'.prev\_id \leftarrow M_{global}(a.prev\_id)$
(7)       Add pair $(a'.id, a.id)$ to $M$
(8)       Add $a'$ to $D$
```

Algorithm Specifics. We combine the method for determining new actions with our relabeling strategy to obtain robust algorithms for incrementally loading from and saving to a database. Algorithm 1 describes the loading algorithm and Algorithm 2 summarizes the saving algorithm. In each algorithm, we use either the database or local version tree to update the other depending on the direction, ensuring that new ids are assigned, existing ids are remapped, and the global-to-local mapping M is updated. Note that all entities are updated in place, copying only the (new) required information from one side to the other.

4.3.2 Implementation

We have implemented the synchronization mechanism on top of the VisTrails system (http://www.vistrails.org). The implementation consists of a client/server architecture shown in Figure 4.2. The server-side is a MySQL database that stores version trees. Users can create synchronization sessions through the user interface (see below). The standard VisTrails database

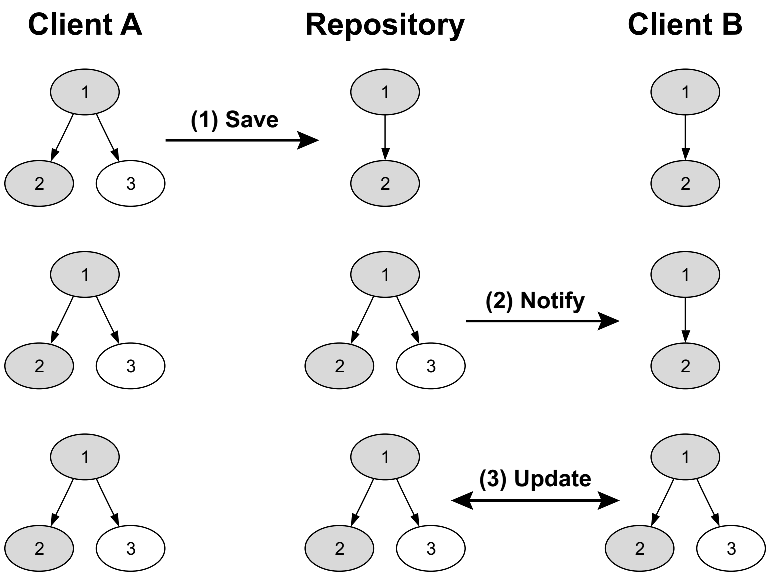


Figure 4.2: The synchronization algorithm. Client A creates a new change (labeled as version 3). This new version is automatically saved to the repository (Step 1). Whenever the repository is updated, it notifies all clients of the new change (Step 2). All clients (including Client B) then incrementally update themselves (Step 3).

schema has been extended to store information about synchronized sessions. This information includes the ids of synchronized version trees, user ids, IP addresses, and port numbers. A database trigger uses this information to notify clients when relevant updates are available. The notification is done by an external MySQL function that uses a socket to connect to the client. The message to the client includes the version tree id number so that the client can request the updates for that version tree. Note that messages about changes to a given version tree are sent to all users using that version tree, except to the user whose changes activated the trigger.

The client-side application is a modified version of VisTrails; the modifications include code for performing incremental updates and saves against the database and for receiving notification messages from the database. Because the system contains a controller object for each version tree, we use it to monitor these notifications and start update procedures. Because the controller is linked to the GUI, we also need to redraw the version tree whenever synchronization modifies the tree.

To setup synchronization, users need to select (or create) a database to serve as a centralized repository. This database must have the schema as outlined above and the synchronization triggers that send the update

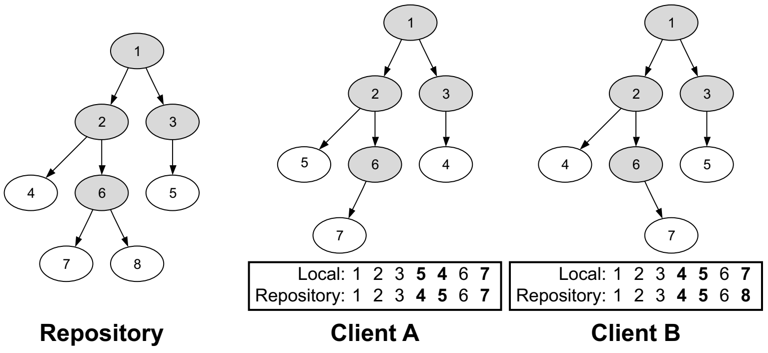


Figure 4.3: Relabeling. Because two users may make updates at the same time or may temporarily lose their connections with the repository, the ids of their nodes may not correspond with the repository's ids. To solve this problem, each client stores the tree according to its own local ids and maintains a map to the repository's global ids.

notifications. Once the database is in place, users connect to it and select the version trees they want to share. After that, the synchronization (sync) mode can be enabled with the push of a button. From that point on, the version tree will be kept in sync with the central repository and the other users. To help distinguish between versions, those created by other users are shown in blue while one's own versions are highlighted in orange.

4.3.3 Issues

Mutable Objects. The monotonicity of the version tree is required for the synchronization process. Change actions and operations are immutable: they are never modified after they are stored in the repository. Thus, the system only needs to check for *new* objects in order to perform synchronization. There are, however, *mutable* objects associated with actions for which this optimization cannot be applied. For example, VisTrails has *version tags* and *version annotations* associated with workflows that can be modified, and these modifications are not saved as actions. Version tags assign text labels to workflow versions while version annotations store general notes about the version. Because changes to these objects are non-monotonic (and destructive), *all* objects must be saved and loaded during each incremental load/save. Locally, we can keep a flag that indicates whether or not the entity changed so that we only need to save it when it does, but the same cannot be done for the global repository. Nonetheless, since the volume of mutable data is small, we copy all instances during an incremental load.
Integrating Changes. A nice feature of our synchronization framework is that it does not require the user to integrate another user's changes. However, consider the situation where two users (A and B) are working on a

similar problem, and they have attacked different pieces of it from a common starting point. Each has seen that the other has made changes, but they wanted to finish their own piece. Later, when they decide to integrate these changes, user A can switch to B's version and make the changes applied in her own version. A more efficient alternative would be for user A to use the *analogies* mechanism [SKV+07] implemented in VisTrails to automatically apply the changes from one branch to another.

Local parameters. Workflows may not always have the same meaning to all users, and they may disagree about certain parameter settings or methods used. For example, an input filename parameter may differ between two users because the users store the file in different disk locations. Currently, the only way to deal with such local parameter settings is to create a different version for each set of parameters. This means that a change in one user workflow will not propagate to the other version, which is not desirable. A solution to this problem could be to separate the shared workflow from the local settings, creating a division of the workflow in some way.

Data sharing. The ability to share data is an important part of collaboration. For workflows, you may want to share output data as well as input and intermediate results. This can be done with a data pool which maintains up-to-date data items created by the users. This would make it possible for users not only to see each other's results, but also use the data as inputs to other workflows. The COVISA project [WWB97] implements this kind of data sharing. Users can exchange data and directly use it in their pipelines. Another system that implements the idea of a data pool is the *Data Playground* [GGW+07]. The Data Playground provides a workflow editor that is highly data-centric, letting users view and import data while they compose workflows that in turn create new data items. This gives the users control over their data while they experiment with different data manipulation operations. The prototype only works for one user but it shows how a data centric view can be used in collaborative workflow design.

Module packages. Users must have a common set of module packages in order to share workflow specifications. Module packages contain sets of modules that perform similar functions, much like web services. If one collaborator is missing a module, a workflow containing that module cannot be executed. For workflows that require many different packages and libraries, an effective mechanism is needed for sharing. This can be done, for example, through the use of public repositories or automatic methods for users to import module packages from other users as they are required. The packages are often platform specific and versioned, so finding the right package is not trivial. This requires packages to use a good versioning scheme, possibly with backward-compatible packages. Different package versions also need to exist for different platforms so that the platform can be switched while maintaining compatible packages. Another way to handle module sharing is to use shared computing infrastructure, such as the TeraGrid (http://www.teragrid.org), which provides a comprehensive set of packages.

4.3.4 Discussion

While there are many systems that provide mechanisms to deal with the difficulties associated with the collaborative modification of files, they are not built to handle structured information like workflows. For this reason, many workflow systems lack comprehensive version control for their workflow specifications.

Many systems have been developed with the singular purpose of providing version control. Software such as SVN [PCSF04], CVS [Ced93], and Visual Source Safe [Mic97] are optimized to robustly handle the version control requirements associated with source code. Unfortunately, when dealing with workflow descriptions, the standard merge operations common to text files are inadequate and require specialized processing. A second issue is that these systems require users to *manually* perform check-ins and check-outs in order to synchronize versions. Finally, users are often required to merge their changes with older changes, making it more difficult to explore new directions.

We address the shortfalls of standard version control with our method based on synchronizing workflow evolution provenance. Using this approach, workflow descriptions can be analyzed and modified to provide a truly multi-user, collaborative environment, in real time. These modifications provide the basis for version control of rapidly evolving, collaboratively created workflows. The intuitive system allows closer collaboration between users by immediately alerting all users of each other's changes.

4.4 Related work

In this chapter we present, to the best of our knowledge, the first proposal for an infrastructure that supports real-time collaborative workflow design. Trying to use knowledge from multiple sources in the creation of workflows is nothing new, but we are unaware of previous work that has addressed this in real-time situations. This section discusses other techniques for enabling collaborative design.

There are existing mechanisms that can be used for collaborative design of workflows. One of the most general and common methods of real-time collaboration is through remote desktops like VNC [RSFWH98]. By using this method in the design of a workflow, one user can see another user perform operations, such as dragging modules around and creating connections. But for more efficient modes of interaction, both users need to have control simultaneously, and be able to choose whether to take notice of other users' activities. In addition, provenance information would be lost, since it is not possible to distinguish changes performed by different users.

A related area is that of collaborative visualization such as the COVISA project [WWB97] and NoCoV [WBHW06]. COVISA enable several modes of collaboration like sharing data, sharing control of parameters and instructor driven collaboration where one user is in control of another user's pipeline. NoCoV enables users to collaboratively edit a pipeline consisting of instances of *Notification Web Services*. Both of these systems enables collaboration in the creation of the visualization pipeline but they do not support the exchange or existence of different versions of the pipeline.

The use of real-time collaboration has been explored in other areas. *Co-browsing* [Ese02] enables multiple people to browse by sharing a web browser view and following links together. Similar to VNC, co-browsing is useful when a user wants to guide another through a browsing session. However, unlike VNC where the whole desktop is shared, in co-browsing users only share a browser view. Co-browsing can thus be more efficient, since only clicks within a browser view need to be propagated to the users.

Sharing workflows through public repositories like myExperiment [mye] and Yahoo! Pipes [Yah] serves other purposes. These repositories foster the re-use of knowledge through searchable sites. However, the synchronization infrastructure we propose could potentially be a useful feature offered by these sites.

4.5 Conclusion

This chapter described an infrastructure that supports real-time collaborative design of workflows. This infrastructure represents a version of the workflow lifecycle in which workflows are subject to constant and immediate reuse. The workflow design knowledge is shared in real time, which can improve close collaborations. As such, it is an essential part of improving workflow reuse.

This infrastructure can be integrated with any workflow system that captures workflow evolution provenance. Our implementation of the synchronization mechanism on top of the VisTrails system shows that workflow systems can be a powerful tool for real-time collaboration. Users can collaborate efficiently and effectively, exploring different branches and taking advantage of each other's progress. Together with techniques for data sharing and remote execution, this enables efficient creation of complex workflows.

By leveraging the concise representation of workflows provided by the change-based provenance model, synchronization is efficient: only incremental changes need to be propagated to collaborating users. However, further experiments are needed to assess the scalability of the current implementation.

We believe that our provenance-based synchronization mechanism can be applied to applications other than workflows. Combined with techniques to visualize provenance information, this mechanism can serve as a powerful platform for collaborative design in general. Users can share their work effectively while inspecting each other's contributions. The application of our synchronization infrastructure in other areas of computational design is a direction we plan to pursue in future work.

Chapter 5

A First Study on Presenting Workflow Search Results

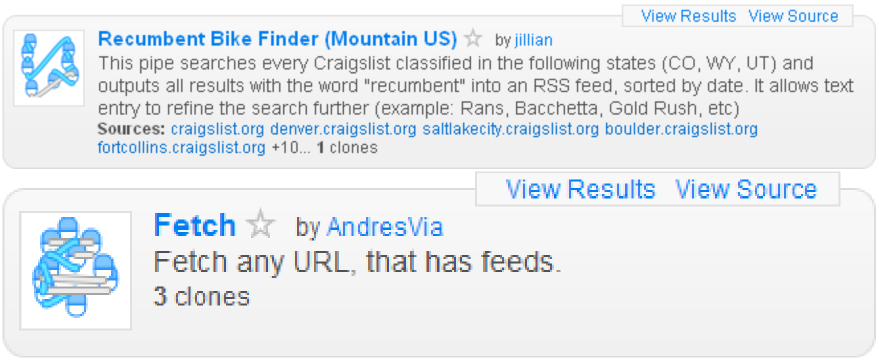
An important aspect of workflow reuse is to be able to search for workflows efficiently. In this chapter, we have conducted an initial study of presenting workflows as search results. The motivation for this work was presented in Chapter 2. The presentation of search results is an important aspect of a workflow search engine, as it is a way for users to evaluate suitable workflows once they have been retrieved by the search engine. This chapter studies alternative techniques for deriving concise, yet informative, workflow snippets. We study both the summarization of individual workflows, and the summarization of workflow sets. The goal is to help users identify relevant workflows without having to look at the full workflow specification.

This work has previously been published as [ESLF09].

5.1 Introduction

There are different pieces of information associated with a workflow, including its specification (modules, connections and parameters) and additional metadata, such as a textual description and the workflow creator. To display search results, an important challenge is how to summarize this information. Workflow snippets are short summarizations of workflows that enable users to select the most promising results without having to access all the details of the workflow itself. As with snippets generated by traditional search engines, workflow snippets can help users more quickly identify *relevant* workflows, without having to inspect their details.

Existing workflow search engines, such as Yahoo! Pipes and myExperiment [mye], use metadata associated with workflows to derive snippets (see



Recumbent Bike Finder (Mountain US) by jillian

This pipe searches every Craigslist classified in the following states (CO, WY, UT) and outputs all results with the word "recumbent" into an RSS feed, sorted by date. It allows text entry to refine the search further (example: Rans, Bacchetta, Gold Rush, etc)

Sources: craigslist.org denver.craigslist.org saltlakecity.craigslist.org boulder.craigslist.org fortcollins.craigslist.org +10... **1** clones

Fetch by AndresVia

Fetch any URL, that has feeds.

3 clones

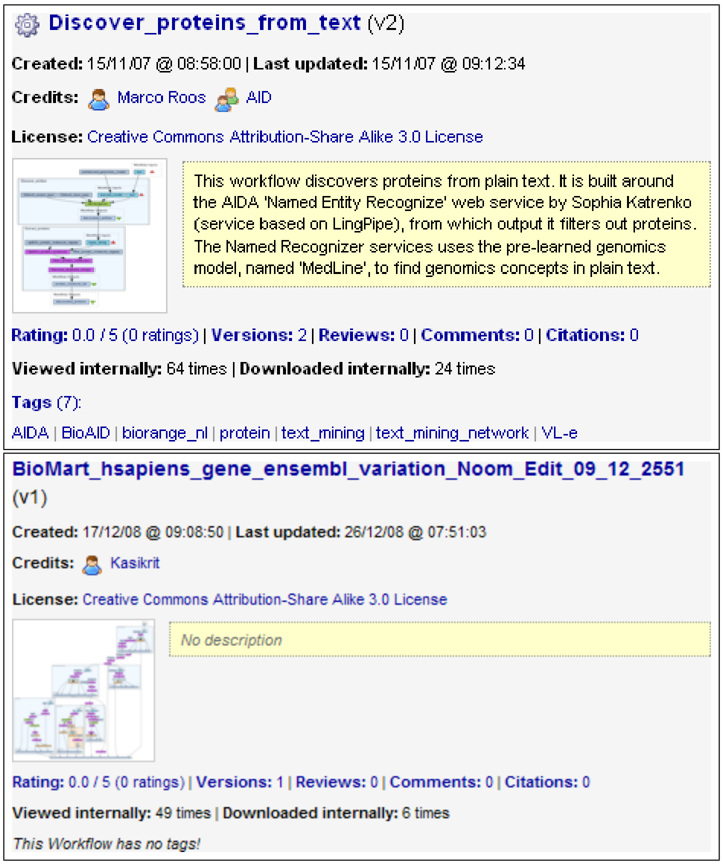
Figure 5.1: Workflow snippets from Yahoo! Pipes. The quality of snippets depends on the quality of the metadata associated with a workflow. The snippet on the top has an informative description, the one on the bottom does not.

Figures 5.1 and 5.2). As a result, the quality of the snippets is highly dependent on the quality of the metadata associated with the workflows. For example, while the top snippet in Figure 5.1 provides a detailed description of the workflow, the one on the bottom does not. Although both snippets show a thumbnail with a visual representation of the workflows, neither exploits the actual workflow specification. We posit that by exploiting the workflow specification, we can generate high-quality snippets even when metadata is poor or non-existent.

An important challenge we need to address is how to display *enough* structural information, given space constraints. We are faced with conflicting requirements: snippets need to be concise and, at the same time, informative. If snippets are too large (see e.g., Figure 5.2), only a few will fit on a page, forcing users to browse through multiple pages. If they are too small, users will be forced to examine a potentially large number of workflows. Since workflows can be complex, contain several modules and many more connections, not only is it hard to fit multiple results on a page, but also the complexity of the workflows can confuse users.

In many cases methods that deal with individual workflows are not efficient on larger sets of workflows. [SKV$^+$07] describes a method to show the differences between pairs of similar workflows, but this method quickly becomes too complex when dealing with more than two workflows. It also requires the number of workflows to be small, and similar. This does not scale in a search engine where a large number of highly dissimilar workflows need to be presented at the same time. Thus we need algorithms that are scalable to large sets of workflows.

Unlike textual documents [TTHW07], the structure present in workflows makes the problem of summarization more challenging. Our strategies take into account that workflows can be large and dissimilar, and that multiple



Discover_proteins_from_text (v2)

Created: 15/11/07 @ 08:58:00 | **Last updated:** 15/11/07 @ 09:12:34

Credits: Marco Roos AID

License: Creative Commons Attribution-Share Alike 3.0 License

This workflow discovers proteins from plain text. It is built around the AIDA 'Named Entity Recognize' web service by Sophia Katrenko (service based on LingPipe), from which output it filters out proteins. The Named Recognizer services uses the pre-learned genomics model, named 'MedLine', to find genomics concepts in plain text.

Rating: 0.0 / 5 (0 ratings) | **Versions:** 2 | **Reviews:** 0 | **Comments:** 0 | **Citations:** 0

Viewed internally: 64 times | **Downloaded internally:** 24 times

Tags (7):

AIDA | BioAID | biorange_nl | protein | text_mining | text_mining_network | VL-e

BioMart_hsapiens_gene_ensembl_variation_Noom_Edit_09_12_2551 (v1)

Created: 17/12/08 @ 09:08:50 | **Last updated:** 26/12/08 @ 07:51:03

Credits: Kasikrit

License: Creative Commons Attribution-Share Alike 3.0 License

No description

Rating: 0.0 / 5 (0 ratings) | **Versions:** 1 | **Reviews:** 0 | **Comments:** 0 | **Citations:** 0

Viewed internally: 49 times | **Downloaded internally:** 6 times

This Workflow has no tags!

Figure 5.2: Workflow snippets from myExperiment. While the top snippet contains a detailed description of the workflow and additional metadata, such as popularity and number of downloads, the bottom snippet contains very little information making it difficult to determine what the workflow actually does.

workflows need to be presented in a limited space. Our contributions can be summarized as follows:

- We discuss requirements for generating high-quality workflow snippets;
- We propose alternative techniques for selecting important subsets of the workflow information to be displayed in a snippet;
- We present a user study where we evaluate the proposed techniques and compare them against existing workflow snippet generation strategies. Our preliminary results show that quality of the snippets (as perceived by the users) improves when structural information is included.

In Section 5.2, we define the problem of generating workflow snippets and discuss the requirements for generating high-quality snippets. We describe different strategies for generating and presenting the snippets in Section 5.3. In Section 5.4, we evaluate the snippet strategies and discuss our preliminary results. We present related work in Section 5.5. We conclude the chapter in Section 5.6.

5.2 Problem Formulation

We will now provide a formal definition of a workflow, and discuss the requirements for generating workflow snippets.

5.2.1 Definitions

The workflow is defined in Section 2.2.1. Figure 5.3A shows a workflow consisting of a connected set of modules. Pre-defined parameters on modules are not shown. Besides the *workflow specification* W, workflows are also associated with a list of annotations A that adds metadata to the workflow. *Annotations* are triples on the form $subject, key, value$ where subject is a component of the workflow (e.g., a module or sub-graph), *key* is the type of annotation, and *value* is the actual annotation. Annotations are used to attach metadata to the workflow.

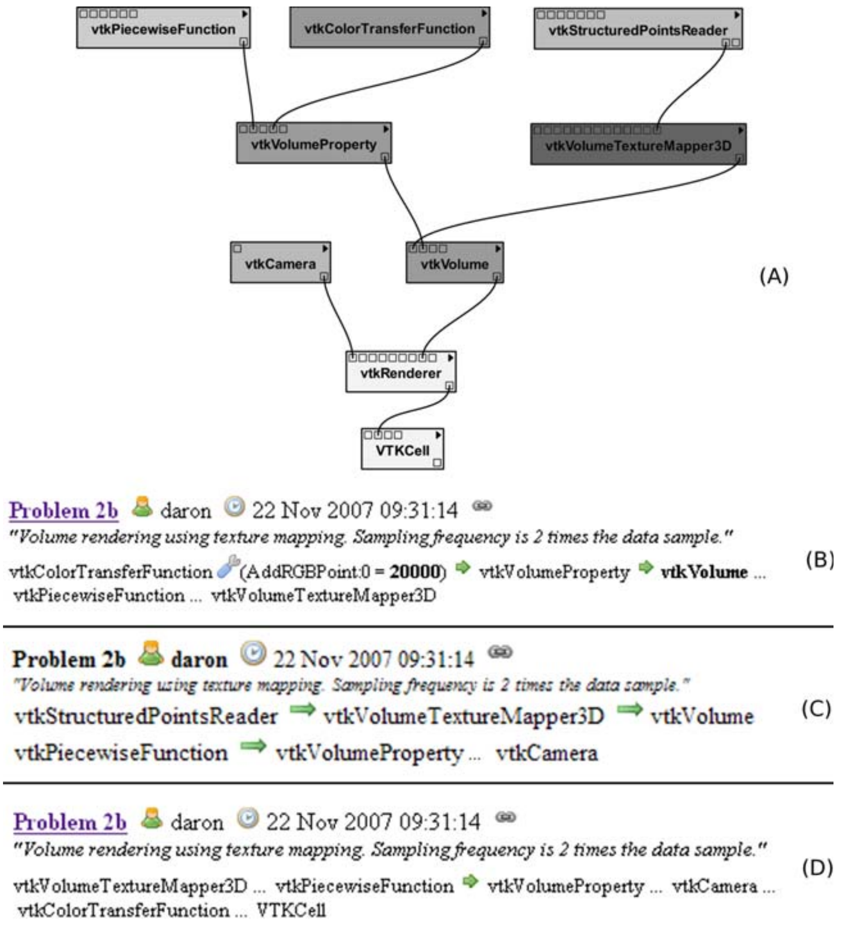
A *workflow collection* C contains a set of workflows. A *search query* consists of a list of query terms Q. Any word w in the workflow specification such that $w \in Q$ is considered a *keyword*.[1] A search query over C returns a *result set* R where $R \subset C$ and $|R| = k$. A *workflow snippet* S provides a summary of a workflow specification. It consists of a *subset of the information in the workflow specification*: $S \subset W \cup A$.

Snippet Generation. Given a workflow w, we aim to construct a snippet S that is informative and concise. The workflow collection (C), query (Q), and result (R) set are potential sources of information that can be included in the snippet. We introduce a function $FScore$: $\{C, Q, R, W, S\} \rightarrow c$, which, given these information sources, the snippet S and workflow W, outputs a score $c \in \mathbb{R}$, indicating the usefulness of the snippet. Our goal is to generate a snippet S with the highest possible $FScore$.

5.2.2 Snippet Requirements

Snippets should allow users to browse and determine the relevance of workflows more efficiently, i.e., a user should be able to identify workflows that are relevant to her query by examining the snippets, and should not need to inspect the details of each workflow in the result set.

[1]Although, in this chapter we focus on keyword-based queries, our techniques can be extended to support structural queries [SKV+07, SVK+08].



Problem 2b daron 22 Nov 2007 09:31:14
"Volume rendering using texture mapping. Sampling frequency is 2 times the data sample."
vtkColorTransferFunction (AddRGBPoint:0 = **20000**) → vtkVolumeProperty → **vtkVolume** ...
vtkPiecewiseFunction ... vtkVolumeTextureMapper3D

Problem 2b **daron** 22 Nov 2007 09:31:14
"Volume rendering using texture mapping. Sampling frequency is 2 times the data sample."
vtkStructuredPointsReader → vtkVolumeTextureMapper3D → vtkVolume
vtkPiecewiseFunction → vtkVolumeProperty ... vtkCamera

Problem 2b daron 22 Nov 2007 09:31:14
"Volume rendering using texture mapping. Sampling frequency is 2 times the data sample."
vtkVolumeTextureMapper3D ... vtkPiecewiseFunction → vtkVolumeProperty ... vtkCamera ...
vtkColorTransferFunction ... VTKCell

Figure 5.3: (A) A workflow where darker modules indicate more uncommon modules (IDF-value). (B) A snippet showing the neighborhood matching the query "vtkVolume AND 20000". The query keywords are highlighted and the keyword neighborhood is displayed as a set of paths. (C) A snippet showing the modules with highest IDF value. (D) A snippet showing representatives for the most important groups.

Huang et al. have studied the problem of generating snippets for XML query results [HLC08]. They outline four criteria to evaluate snippets. Snippets should be: *self-contained*, *representative*, *distinguishable*, and *small*. Below we describe how we map these criteria to the workflow snippets, and in Section 5.3, we discuss how these criteria are used as the basis to design snippet generation strategies.

Self-contained. A snippet should show the context of keywords in the query. Each module m is associated with a set of keywords $key(m)$ which are extracted from the module name, type, annotations, and parameters. If a query Q matches a module, $key(m) \cap Q \neq \emptyset$, the module m and other modules in its neighborhood should be included in the snippet. Since there can be overlaps of the neighborhoods of modules that match a query, the snippet generation strategy must find the set of modules $M_S \subset M$ that is *most relevant*.

Representative. A snippet should capture the essence of each workflow—what the workflow actually does. This is analogous to how sentences are selected to represent a document in text snippets. We need to identify the most prominent features in the workflow graph that best represent its semantics, i.e., $M_R \subset M$.

Distinguishable. The difference between any pair of snippets should be visible. To ensure this, we must find and display the structural differences among the workflows in a result set. In other words, we need to identify the set of most distinguishing modules $M_D \subset M$. This is analogous to clustering text documents and extends the usual representation of snippets to also include information about other workflows in the result set.

Small. Snippets should be compact so that many can fit on a single result page. We do this by fixing the maximum number of modules in each snippet to g.

Given these requirements, a workflow snippet can be constructed as a combination of the relevant modules $S = \{M_S \cup M_R \cup M_D\}$, such that $|S| \leq g$. To understand the implications of applying these different criteria, we will examine each individually below. We note, however, that in the general case we need a mechanism to weigh the different measures to be able to select the *best* combination of modules.

5.3 Snippet Generation

In this section we address the problem of presenting snippets for single workflows. In the introduction we argued that workflow snippets need to exploit the graph structure of workflows. To concisely represent a workflow graph, we propose different summarization strategies that selectively display (or hide) components of workflow graphs. Before presenting different strategies for generating and displaying snippets, we describe two metrics that capture the notion of importance of sub-components of a workflow.

5.3.1 Structural Importance

In order to select the subsets of a workflow that should be displayed (or hidden) in a snippet, we first need to define a notion of importance. The intuition behind our choice comes from two empirical observations: modules that are present in a given workflow but that are unusual in the result set (or in the workflow collection) might be of interest to the user—they are a *distinguishing* feature of the workflow. Furthermore, modules that occur together in many workflows represent a semantic entity (a pattern) that can be of interest to the users.

To identify distinguishing modules we apply a measure that is widely used in information retrieval: *Inverse Document Frequency* [BYRN99]. IDF is defined as:

$$idf(m) = \log N/N(m)$$

where $N = |C|$ and $N(m) = |\forall W : W \in C \wedge m \in W|$. We can then estimate, for each module type, how rare it is in the collection. For example, Figure 5.3A shows a workflow where darker modules correspond to modules that have a higher IDF value—the modules are more uncommon in the collection.

By measuring module co-occurrence (or semantic proximity), we can find interesting groups of modules. These groups form semantic entities that can be represented in a more compact way, e.g., by collapsing a group into a single module. A similarity measure that fits our requirements is the *Jaccard distance* [VR79]. The semantic distance between module types M_A and M_B can be defined as:

$$dist(M_A, M_B) = 1 - \frac{P(M_A \cap M_B)}{P(M_A) + P(M_B) - P(M_A \cap M_B)}$$

which requires the frequency of M_A in C, the frequency of M_B in C, and the frequency of M_A and M_B occurring together in C. Note that $dist(M_A, M_B) = 0$ when M_A and M_B always occur together, and $dist(M_A, M_B) = 1$ when they never occur together.

5.3.2 Module Selection Strategies

We will now propose four different snippet generation strategies. Our goal is to highlight structural features of the workflows, and each strategy uses a different method to select which modules in the workflow to include in the snippet.

Query-Neighborhood Strategy. This strategy identifies important modules M_S in the neighborhood of a set of modules that match the query keywords. We use the following analogy to snippets for text documents: modules represent words, and connected sets of modules correspond to sentences. We first choose all modules M_Q that match the query Q, and traverse their connections to find the neighborhood. This can be done through a breadth-first search from the initial set of modules that stops when g modules are found. Since g is small, we select the modules with the highest IDF value first, i.e., the most descriptive modules. The snippet in Figure 5.3B shows a representation of such a neighborhood split up into paths.

IDF Strategy. The goal of this strategy is to discover a set M_R of modules that are *representative* for a workflow. Using the IDF measure, we can choose the top g modules with the largest IDF value. This strategy is

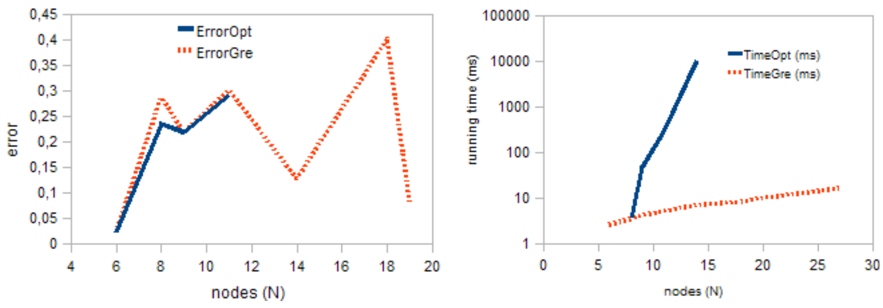


Figure 5.4: Comparison between optimal and greedy grouping algorithms. Left, difference in total error versus number of nodes. Right, the running time versus number of nodes.

used for the snippet in Figure 5.3C. The main advantage of this strategy is that common modules that do not contribute to the understanding of the workflow are selected last. A possible drawback is that the most uncommon modules may occur within the same part of the workflow, which may leave other, potentially important sub-graphs unrepresented. The next strategy addresses this issue.

Grouping Strategy. This strategy attempts to group together modules that co-occur often. This is based on the observation that workflows may contain sub-graphs with specific functionality that can be presented as a single entity, making for a more compact and yet useful representation. We divide the graph into sub-graphs (groups) such that each group M_n is a connected sub-graph of M. We want to find a set of disjoint groups $G = \{M_1 ... M_g\}$ that covers the whole graph. We use the Jaccard distance to measure the semantic relatedness of specific groups using:

$$MScore(M_n) = \frac{\sum\limits_{m_i, m_j \in M_n} dist(m_i m_j)}{|M_n|}$$

The score for the workflow is:

$$GScore(G) = \sum_{M_i \in G} MScore(M_i)$$

Testing all possible G reduces to the *Exact Cover* problem [Kar72], which is NP-complete. We have used a greedy algorithm that starts with a group for each module, and merges the two adjacent groups that give the best *GScore* until g groups remain. This approximation reduces the complexity to polynomial time. Figure 5.4 shows the difference between optimal and greedy versions. The optimal version fails when $|M| > 14$ whereas the greedy scales well. The approximation error of the greedy version is small in most cases, but was not computable for large workflows.

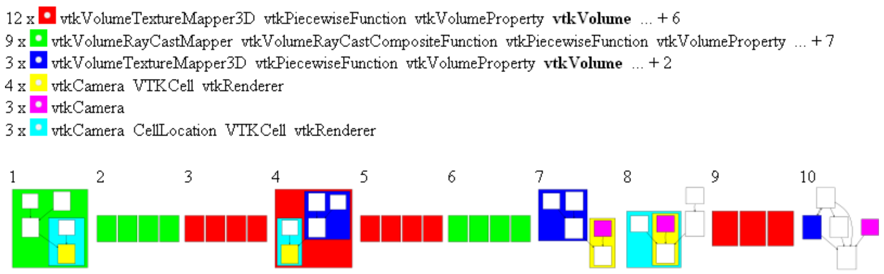

12 x 🟥 vtkVolumeTextureMapper3D vtkPiecewiseFunction vtkVolumeProperty **vtkVolume** ... + 6
9 x 🟩 vtkVolumeRayCastMapper vtkVolumeRayCastCompositeFunction vtkPiecewiseFunction vtkVolumeProperty ... + 7
3 x 🟦 vtkVolumeTextureMapper3D vtkPiecewiseFunction vtkVolumeProperty **vtkVolume** ... + 2
4 x 🟨 vtkCamera VTKCell vtkRenderer
3 x 🟪 vtkCamera
3 x 🟦 vtkCamera CellLocation VTKCell vtkRenderer
1
2
3
4
5
6
7
8
9
10

Figure 5.5: Multiple snippets having common groups. The algorithm identifies shared groups and shows them with colors. A legend showing the meaning of the colors is shown on the top. (Best viewed in color.)

Once we have a set of groups, for each group G, we select the module with the highest IDF in G to be the representative for that group. Another alternative would be to select the module with the lowest Jaccard distance, but this leads to the selection of modules that are too common and do not reflect the complexity of the group. The IDF method, in contrast, often selects uncommon (more specialized) modules and provides more meaningful labels for the groups. Figure 5.3D shows an example of a snippet derived by this strategy. Although the groups are not shown explicitly, notice the addition of $VTKCell$ that represents a previously unrepresented part of the workflow.

Difference Highlighting Strategy. This strategy aims to display the differences and similarities among workflows in a result set. Because workflows in a set may share sub-graphs, seeing the differences and similarities may help users obtain a global understanding of the workflows and identify specific features (e.g., modules or groups that occur in many workflows). Since a result set consists of multiple workflows, it is not possible (or even desirable) to show *all* differences, so we focus on identifying the most prominent ones. The strategy works as follows. First, the grouping strategy is used to find common groups. To increase the number of matches, we consider two groups if they contain the same types of modules, disregarding the graph structure. Sub-groups can then be selected that makes it possible to match sub-graphs of large workflows with small workflows. This attenuates the effect of oversimplifying large workflows, which leads to the hiding of structures that they may have in common with smaller workflows. Based on user feedback (see Section 5.4), we have designed a presentation method for highlighting the group containment, which is illustrated in Figure 5.5. The snippets have been aligned horizontally to make the differences clearer.

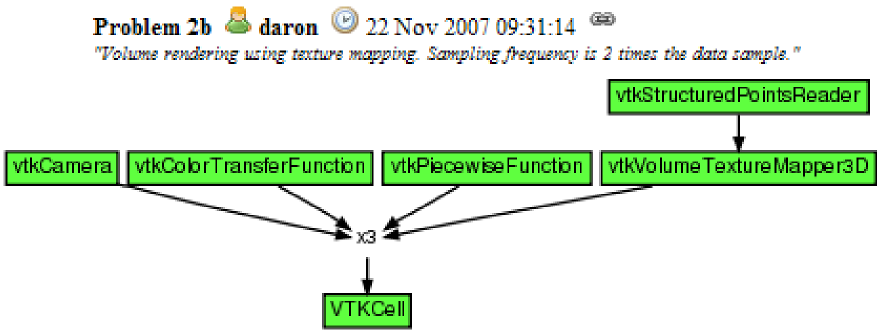


Figure 5.6: A snippet presented as a partial workflow graph. $x3$ indicates that 3 modules have been collapsed.

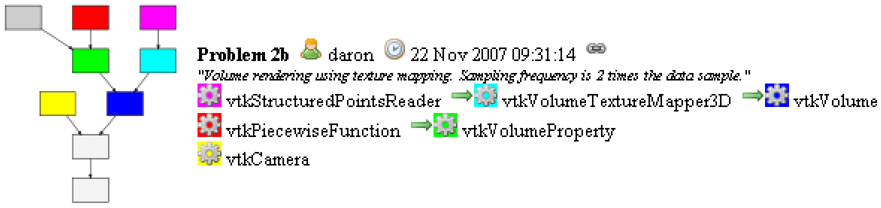
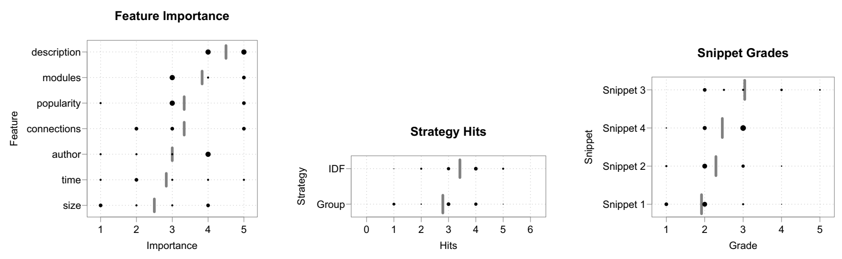


Figure 5.7: A snippet presented as a graph with a color legend describing the modules. (Best viewed in color.)

5.3.3 Snippet Presentation

As Figure 5.3 illustrates, a textual representation for a snippet can be compact and informative, but structural information is lost since only a few paths of the workflow are shown. As an improvement, we propose presenting a dynamic image of a sub-graph of the workflow that represents the thumbnail, as shown in Figure 5.6, which uses the same strategy as the snippet in Figure 5.3C. The idea is to make the graph similar to that seen in a workflow editor and thus easier to interpret for experienced users. The disadvantage is that processing time increases—multiple graphs must be dynamically rendered, and additional screen space is needed for each snippet. As a compromise, we designed a snippet where a textual legend is displayed next to the graph (see Figure 5.7). The graph can then be very small and still contain dynamic information. A problem seen with this approach is that users must keep referring to the legend to understand the graph.

When presenting differences between multiple snippets, the user feedback suggested that we should not represent each snippet individually, but together, and clearly highlight their differences and similarities. This calls for a more compact way of presenting multiple snippets. An alternative



(a) (b) (c)

Figure 5.8: Results of questionnaire. Gray bars represents mean scores. Black circles represent observed scores: larger black circles means more scores on that value. (a) Features that users found important sorted by mean importance. (b) The scores for the IDF and Grouping strategy. (c) User grades for the four snippets sorted by mean value.

approach is to highlight common parts in each workflow, as shown in Figure 5.5. Groups that are similar have the same color and differences are represented by white groups. It is then possible to see if workflows are similar in structure. One drawback of this method is that users must refer to the legend to know what each group represents, and groups without a legend are not useful. It is also sometimes not intuitive to spot the differences between workflows; they may have nothing in common; the group order may be rearranged, breaking spatial similarity; and some differences may be hidden, due to lack of space.

5.4 Preliminary Evaluation

In order to evaluate the proposed snippet generation strategies, we performed a user study where we sought both qualitative and quantitative feedback. In our study, we looked at workflows created by students of a visualization course, which make use of the VTK library [Kit].

Quantitative Feedback. In order to examine the effectiveness of our methods we designed a questionnaire and applied it to six subjects: three were experts and three were knowledgeable users. The questionnaire consisted of three parts. The first part asked users to score different workflow features. The results are presented in Figure 5.8(a). The workflow *description* was seen as most important component, followed by the *module* types in the workflow. This supports our intuition that what the workflow does is important, whether it is described (in text) or presented by showing workflow specification (the graph). The actual structure, i.e., *connectivity* and *size*, seems less important. Metadata like popularity, *author* of the workflow, and creation *time* were seen as somewhat important, but orthogonal to contents.

In the second part users were presented with four workflows which were chosen to reflect differences in size and structure. Each user first chose the six most important modules in each workflow. Then we counted how many module types users selected that were common to the modules selected using the IDF and grouping strategies. The resulting scores are shown in Figure 5.8(b). The mean score for IDF is 3.4 and the core for grouping is 2.8. The scores are relatively low since not all users selected the same modules. Nonetheless, these scores indicate that IDF is a good measure of importance with the grouping strategy slightly behind. In the second task, users were asked to score four different snippet types: Snippet 1 contained the same information as the Yahoo! Pipes snippet in Figure 5.1, showing a thumbnail of the workflow; Snippet 2 contained structure presented as text as in Figure 5.3; Snippet 3 contained thumbnail with legend as in Figure 5.7; and Snippet 4 contained the workflow sub-graph as in Figure 5.6. The results are presented in Figure 5.8(c). Snippet 3 scores best and also contains the most information. Snippet 4 comes second, indicating that it may not be worthwile to have a bigger snippet at a higher cost. Snippet 2 scores better than Snippet 1, indicating that users prefer to have structural information in the snippet.

We note that, overall, the scores seem low. This reflects the scale we used, where 3 = *good* and 5 = *excellent*. But it also suggests that there is room for improvement.

Qualitative Feedback. Preliminary user feedback gave valuable insights. The users found that showing the neighborhood of the query keyword and using IDF to find important modules led to better snippets. The labeling strategy, on the other hand, was criticized because it splits the structure of the workflow and the contents of the modules. In general, users found it inconvenient to see groupings that lacked labels and whose structure greatly differed from the structure of the original workflow (see Figure 5.5).

5.5 Related Work

Current workflow search engines such as Yahoo! Pipes and myExperiment use descriptions and coarse-grained thumbnails in their snippets. Our work extends this to consider fine-grained structural information. The WISE [SSC09] workflow search engine addresses the related problem of showing sub-workflows containing keyword terms. Altintas et al. have worked on creating reusable components for grid workflows [ABB^{+}05]. Grid workflows control high performance computational pipelines and include authentication, job scheduling, and data replication. Many different technologies are used which makes the workflows very specific and hard to reuse. In their work they show how components can be abstracted into tasks that are independent of the underlying technology. These components represent common operations that are easier to understand and reuse. Such components are an essential first step in making workflows reusable.

Structural snippets have been explored for XML documents [HLC08], where Huang et al. propose a set of requirements for snippets that we used as the basis for designing workflow snippet generation strategies. Previous works on snippets for ontologies on the Semantic Web uses semantic proximity of concepts computed with different tools and databases such as WordNet[2] to calculate the query neighborhood [PWTY08]. They focus on self-similarity and keyword proximity and have a number of important measures that could be useful for workflow snippets and we plan to consider this in future work. Considerable work exists on finding the query neighborhood for text documents [TTHW07]. Our method is analogous but is applied to the workflow graph. Our work was inspired, in part, by approaches that compress network graphs [GL04]. Although these take into account mainly the topology of the graph, they can be combined with workflow heuristics. Computing the maximum common sub-graph between any two workflow graphs is a related problem [SKV$^+$07], but not directly applicable for snippet generation due to its computational complexity. The problem can be simplified by reducing the workflow structure to a set of module labels, or a multiset of labels. This simplification has been shown to yield a good approximation of the workflow structure [SLA$^+$08]. We use this idea in our group similarity algorithm. Our grouping approach is related to document snippet clustering [ZHC$^+$04] but creates groups as a connected sub-graph. Grouping can be compared with agglomerative clustering, whereas our optimal approach is similar to k-means.

Zhuge [Zhu02] describes a process-matching approach where workflows are matched based on the similarity of the operation types. Concept distances in a predefined process ontology are used to calculate similarity between workflows. Our workflow similarity approach does not use a process ontology, and it is not clear what performance gain and overhead this method has.

5.6 Conclusions

This chapter presents a first study on constructing workflow snippets using information from the workflow graph. Our preliminary results show that structural information is useful and conveys important information about the workflow. Our work hints that the simple IDF strategy is preferable to the more involved grouping strategy and that snippets that resemble the actual workflows are preferred by the users. The results also show that there is room for improvement in the quality of workflow snippets, and we plan to investigate additional strategies in future work.

User feedback showed the need for a better summarizations of sets of workflows. We have begun work on a workflow presentation framework where it is possible to try out different presentation strategies. Our framework is currently being used to explore how similarities between workflows

[2]http://wordnet.princeton.edu

can be presented to users. This is ongoing work and part of future research.

Chapter 6

Conclusions and Future Work

This chapter discusses the contributions in this thesis and outlines the directions for future work. Section 6.1 summarizes the contributions and Section 6.2 and discusses the most relevant directions for future work.

6.1 Summary

This section discusses the contributions as presented in Chapters 3 to 5.

In *Chapter 3* we propose an approach for integrating data provenance. This approach tries to overcome the shortcomings of the data warehousing approach: to convert all data provenance to a common provenance model. Our approach uses the data mediator, which enables access to different provenance models without the need to convert all data. We show the feasibility and scalability of the approach and also our experiences of a case study. The contributions are as follows:

- A global provenance model and a provenance query API showing that it is possible to create a global provenance model and convert queries between different provenance models.
- A description of bottlenecks of transitive closure queries in current data storage models. This shows that efficient provenance queries put requirements on the provenance storage methods.
- We have identified a need for a unified data identifier model for integrating data provenance. Without such a model, global identifiers and namespaces are hard to define.
- We have identified a need for a high level query language for provenance and workflows. To specify queries efficiently we need to be able to construct queries without using model-specific constructs.

Our case study shows that a data mediator can be used as a scalable approach to provenance integration. For small data sizes it is more efficient to convert the data to a common model before querying, but for larger data sizes the mediator model is required to be able to query multiple sources simultaneously. The different provenance models share only a small set of core concepts, which makes such an approach simple to implement.

The case study also highlights some difficulties. The most common types of provenance queries are generally recursive, e.g., searching for a specific record at an unknown depth in a provenance hierarchy. Such queries would require an unknown number of joins in a SQL database. To efficiently perform such queries it is often necessary to rely on implementation-specific methods, which are not available in all cases. In practice, these kinds of queries should be supported in any large-scale provenance store, so it may not be an issue in practice.

When using data from different storage models, it is necessary to make sure that the data items have unique identifiers. In general this is not the case and identifiers will need to be made unique to avoid name clashes. This is a general problem in data integration [ZGS06]. The existing workflow systems can be encouraged to use global identifiers, but there are also competing naming schemes available. In this regard, data identifier name clashes will remain an issue, and need to be addressed in integration approaches.

The query API in the case study is built on procedural methods to construct queries. This worked well since the provenance queries we used were relatively simple. For more complex queries and for better performance it will be necessary to develop a high-level declarative query language. This is discussed in future work.

Chapter 4 proposes a method for real-time collaborative workflow design. This method enables collaborators to work simultaneously on the same workflow. This is made possible by using the workflow evolution provenance to maintain different versions of the workflow simultaneously. We also show how different workflow branches can be merged and how the evolution provenance are distributed in real time between the collaborators. The contributions consists of the following:

- A specification of a method that uses workflow design provenance to make it possible for multiple users to edit a workflow simultaneously.
- A reference implementation showing how the method can be integrated in a workflow editor.

The usage of the workflow evolution provenance is a new approach to collaborative workflow design. This provides new possibilities for people working together. The general idea is simple, and can be extended to provide additional functionality. The mechanisms for fetching and updating the data can be further optimized and the interface can be extended to provide more feedback when updates are fetched.

Chapter 5 proposes new strategies for presenting workflows as search results. We identify the summarization of workflows as an important problem in workflow search and propose the use of structural information to create workflow snippets. The contributions are as follows:

- A specification of requirements for generating workflow snippets which takes into account the structure of the workflow.
- A proposal for new workflow snippet generation strategies based on structural information.
- An evaluation of the snippet generation strategies that shows a preference for IDF-based ranking of structural importance in workflows.

6.2 Future Work

The previous section suggests many directions for future work. This section will discuss the most important of these, which are:

- Data provenance interoperability – The ability to seamlessly integrate data provenance between workflow systems. This is the essential part of collaboration using provenance.
- Query Languages for Workflows and Provenance – Development of expressive query languages that support the needs for workflows and provenance.
- Presenting Workflow Search Results – Develop methods for presenting global properties of workflow collections. This is an essential part of workflow search engines and workflow reuse.
- Workflow Ranking – Improve the ranking for workflows by using structural information. This is another essential part of workflow search engines and largely unexplored for structured workflow queries.

6.2.1 Data Provenance Interoperability

Data provenance interoperability is an essential topic in this thesis. We explored a mediation approach to provenance ineroperability and identified a number of issues related to scalability of our method. To fully evaluate the scalability of our solution it will be necessary to find suitable real-world scenarios and data. One suitable source of data is the Open Provenance Model (OPM) project [MFF$^+$08], which is described in Section 3.2.3. OPM is currently being developed as part of the Provenance Challenge[1]. It is designed as a lightweight and general provenance model able to express different kinds of provenance. OPM has many collaborators, who contribute

[1]More information can be found at: http://twiki.ipaw.info/bin/view/Challenge/

data provenance that can be used by other project members. It will be possible to use these data sets to further investigate the scalability of our mediation approach.

6.2.2 Query Languages for Workflows and Provenance

We would like to explore usable query languages and interfaces for querying data provenance. We have developed a provenance query language (described in Chapter 3), with a query API uses low-level constructs which makes it complex for end-users to specify queries. It can, nonetheless, be used as the basis for high-level query languages and interfaces. Specifically, we plan to develop automatic mechanisms for translating high-level provenance queries into calls to the query API, for example: queries defined through a query-by-example interface [SKV+07]; and through domain-specific provenance languages [SKS+08]. In Section 2.3 we show that models for provenance and workflows are based on the same concepts, and similar query languages can be used in both models. This means our provenance query language can also be applied to workflow models, and used in workflow search engines. Currently we have only used keyword search in our workflow search engine, but enabling structured workflow queries can be achieved by using our provenance query language.

6.2.3 Presenting Workflow Search Results

In Chapter 5 we explored strategies for presenting workflow search results. We presented strategies for displaying summarizations of workflows using structural information. An important next step is to integrate the snippet generation strategies into a workflow search engine and evaluate it in a realistic scenario. This will enable an efficient means of browsing workflow collections, which is an essential part of a workflow search engine.

6.2.4 Workflow Ranking

Workflow ranking is the process of determining the relevance of a workflow to a user query. It is important to have a ranking that captures the user's intent and is able to retrieve relevant workflows. In Chapter 5 we ranked workflows by indexing the module names, descriptions, title, parameters; *etc.*; and using TF/IDF weighting on the terms to compute the ranking. This works well for keyword search, but it needs to be studied for structured queries, since the structure of the query will need to be taken into account when doing the ranking. Ranking, together with the presentation of search results, are the two essential parts of a workflow search engine. Once they work together, they can be a great tool for workflow reuse and scientific collaboration.

Bibliography

[ABB$^+$05] I. Altintas, A. Birnbaum, K.K. Baldridge, W. Sudholt, M. Miller, C. Amoreira, Y. Potier, and B. Ludaescher. A framework for the design and reuse of grid workflows. *Lecture Notes in Computer Science*, 3458:120–133, 2005.

[App] Apple's Mac OS X Automator. `http://www.apple.com/downloads/macosx/automator`.

[Aßm03] U. Aßmann. *Invasive software composition.* Springer-Verlag New York Inc, 2003.

[BD08] Roger S. Barga and Luciano A. Digiampietri. Automatic capture and efficient storage of escience experiment provenance. *Concurrency and Computation: Practice and Experience*, 20(5):419–429, 2008.

[Bea01] D.B. Beaver. Reflections on scientific collaboration (and its study): past, present, and future. *Scientometrics*, 52(3):365–377, 2001.

[BEKM06] Catriel Beeri, Anat Eyal, Simon Kamenkovich, and Tova Milo. Querying business processes. In *VLDB*, pages 343–354, 2006.

[BL97] David M. Beazley and Peter S. Lomdahl. Building flexible large-scale scientific computing applications with scripting languages. In *PPSC*. SIAM, 1997.

[BYRN99] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press/Addison-Wesley, 1999.

[Ced93] Per Cederqvist et al. *Version Management with CVS (for CVS 1.11.6)*, 1993.

[CFH$^+$08] Ben Clifford, Ian Foster, Mihael Hategan, Tiberiu Stef-Praun, Michael Wilde, and Yong Zhao. Tracking provenance in a virtual data grid. *Concurrency and Computation: Practice and Experience*, 20(5):565–575, 2008.

[CFK$^+$00] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23(3):187–200, 2000.

[DF08] Susan B. Davidson and Juliana Freire. Provenance and scientific workflows: challenges and opportunities. In *Proc. of ACM SIGMOD*, pages 1345–1350, 2008.

[DSS$^+$05] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G. Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, and Daniel S. Katz. Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems. *Scientific Programming Journal*, 13(3):219–237, 2005.

[EKA$^+$08] Tommy Ellkvist, David Koop, Erik W. Anderson, Juliana Freire, and Cláudio Silva. Using provenance to support real-time collaborative design of workflows. In *Provenance and Annotation of Data and Processes: Second International Provenance and Annotation Workshop, IPAW 2008, Salt Lake City, UT, USA, June 17-18, 2008. Revised Selected Papers*, pages 266–279, Berlin, Heidelberg, 2008. Springer-Verlag.

[EKF$^+$09] Tommy Ellkvist, David Koop, Juliana Freire, Cláudio Silva, and Lena Strömbäck. Using mediation to achieve provenance interoperability. *Services, IEEE Congress on*, pages 291–298, 2009.

[Ese02] A. Esenther. Instant co-browsing: Lightweight real-time collaborative web browsing, 2002.

[ESLF09] Tommy Ellkvist, Lena Strömbäck, Lauro Didier Lins, and Juliana Freire. A first study on strategies for generating workflow snippets. In *KEYS '09: Proceedings of the First International Workshop on Keyword Search on Structured Data*, pages 15–20, New York, NY, USA, 2009. ACM Press.

[FF95] William B. Frakes and Christopher J. Fox. Sixteen questions about software reuse. *Commun. ACM*, 38(6):75–ff., 1995.

[FKSS08] Juliana Freire, David Koop, Emanuele Santos, and Cláudio T. Silva. Provenance for computational tasks: A survey. *Computing in Science and Engineering*, 10(3):11–21, 2008.

[Fos03] I. Foster. The virtual data grid: a new model and architecture for data-intensive collaboration. In *Proceedings of the 15th international conference on Scientific and statistical database* *management table of contents*, pages 11–11. IEEE Computer Society Washington, DC, USA, 2003.

[FSC+06] J. Freire, C. T. Silva, S. P. Callahan, E. Santos, C. E. Scheidegger, and H. T. Vo. Managing rapidly-evolving scientific workflows. In *International Provenance and Annotation Workshop (IPAW)*, LNCS 4145, pages 10–18, 2006. Invited paper.

[FVWZ02] I. Foster, J. Voeckler, M. Wilde, and Y. Zhao. Chimera: A virtual data system for representing, querying and automating data derivation. In *Proc. of the Conference on Scientific and Statistical Database Management*, pages 37–46, 2002.

[G+07] Dennis Gannon et al. A Workshop on Scientific and Scholarly Workflow Cyberinfrastructure: Improving Interoperability, Sustainability and Platform Convergence in Scientific And Scholarly Workflow. Technical report, NSF and Mellon Foundation, 2007. `https://spaces.internet2.edu/display/SciSchWorkflow`.

[GDE+07a] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers. Examining the challenges of scientific workflows. *Computer*, 40(12):24–32, Dec. 2007.

[GDE+07b] Yolanda Gil, Ewa Deelman, Mark Ellisman, Thomas Fahringer, Geoffrey Fox, Dennis Gannon, Carole Goble, Miron Livny, Luc Moreau, and Jim Myers. Examining the challenges of scientific workflows. *Computer*, 40(12):24–32, 2007.

[GGW+07] Andrew Gibson, Matthew Gamble, Katy Wolstencroft, Tom Oinn, and Carole Goble. The data playground: An intuitive workflow specification environment. In *E-SCIENCE '07: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*, pages 59–68, Washington, DC, USA, 2007. IEEE Computer Society.

[GH08] Jennifer Golbeck and James Hendler. A semantic web approach to tracking provenance in scientific workflows. *Concurrency and Computation: Practice and Experience*, 20(5):431–439, 2008.

[GJM+06] P. Groth, S. Jiang, S. Miles, S. Munroe, V. Tan, S. Tsasakou, and L. Moreau. An architecture for provenance systems. Technical report, ECS, University of Southampton, 2006.

[GL04] A. C. Gilbert and K. Levchenko. Compressing network graphs. In *LinkKDD workshop*, August 2004.

[GMM05] Paul Groth, Simon Miles, and Luc Moreau. Preserv: Provenance recording for services. In *Proceedings of the UK OST e-Science Fourth All Hands Meeting (AHM05)*, September 2005.

[GR07] Carole Anne Goble and David Charles De Roure. myexperiment: social networking for workflow-using e-scientists. In *WORKS '07: Proceedings of the 2nd workshop on Workflows in support of large-scale science*, pages 1–2, New York, NY, USA, 2007. ACM.

[GSLG05] Antoon Goderis, Ulrike Sattler, Phillip Lord, and Carole Goble. Seven Bottlenecks to Workflow Reuse and Repurposing. *The Semantic Web - ISWC 2005*, 3729:323–337, 2005.

[Hal03] Alon Y. Halevy. Data integration: A status report. In *BTW*, pages 24–29, 2003.

[HLC08] Yu Huang, Ziyang Liu, and Yi Chen. Query biased snippet generation in XML search. In *SIGMOD*, pages 315–326, 2008.

[HT03] AJG Hey and AE Trefethen. The data deluge: an e-science perspective. *Grid computing-making the global infrastructure a reality*, pages 809–824, 2003.

[HT05] T. Hey and A.E. Trefethen. Cyberinfrastructure for e-Science, 2005.

[IR88] Yannis E. Ioannidis and Raghu Ramakrishnan. Efficient transitive closure algorithms. In *VLDB*, pages 382–394, 1988.

[Kar72] RM Karp. Reducibility among combinatorial problems, 85-103. In *Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, NY*, 1972.

[KDG+08] Jihie Kim, Ewa Deelman, Yolanda Gil, Gaurang Mehta, and Varun Ratnakar. Provenance trails in the wings/pegasus system. *Concurrency and Computation: Practice and Experience*, 20(5):587–597, 2008.

[Kep] The Kepler Project. `http://kepler-project.org`.

[Kit] Kitware. The visualization toolkit (VTK). http://www.kitware.com.

[KSC+08] David Koop, Carlos Scheidegger, Steven Callahan, Juliana Freire, and Cláudio Silva. Viscomplete: Automating suggestions for visualization pipelines. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1691–1698, 2008.

[LPS92] Terttu Luukkonen, Olle Persson, and Gunnar Sivertsen. Understanding patterns of international scientific collaboration. *Science, Technology, & Human Values*, 17(1):101–126, 1992.

[MF06] Luc Moreau and Ian T. Foster, editors. *Provenance and Annotation of Data, International Provenance and Annotation Workshop, IPAW 2006, Chicago, IL, USA, May 3-5, 2006, Revised Selected Papers*, volume 4145 of *Lecture Notes in Computer Science*. Springer, 2006.

[MFF$^+$07] Luc Moreau, Juliana Freire, Joe Futrelle, Robert McGrath, Jim Myers, and Patrick Paulson. The open provenance model, December 2007. `http://eprints.ecs.soton.ac.uk/14979`.

[MFF$^+$08] Luc Moreau, Juliana Freire, Joe Futrelle, Robert E. McGrath, Jim Myers, and Patrick Paulson. The Open Provenance Model, `http://eprints.ecs.soton.ac.uk/14979/1/opm.pdf`, 2008.

[Mic] Microsoft Workflow Foundation. `http://msdn2.microsoft.com/en-us/netframework/aa663322.aspx`.

[Mic97] Microsoft Corporation. Managing projects with Visual SourceSafe. Redmond, Washington, 1997.

[Mor08] Luc Moreau, editor. *Concurrency and Computation: Practice and Experience– Special Issue on the First Provenance Challenge*, 2008.

[mye] myExperiment. `http://www.myexperiment.org`.

[PCSF04] Michael C. Pilato, Ben Collins-Sussman, and Brian W. Fitzpatrick. *Version Control with Subversion.* O'Reilly Media, Inc., June 2004.

[PRO07] Second provenance challenge. `http://twiki.ipaw.info/bin/view/Challenge/SecondProvenanceChallenge`, 2007. J. Freire, S. Miles, and L. Moreau (organizers).

[PWTY08] Thomas Penin, Haofen Wang, Duc T. Tran, and Yong Yu. Snippet generation for semantic web search engines. In *ASWC*, DEC 2008.

[RDF08] Resource Description Framework (RDF), `http://www.w3.org/RDF/`, 2008.

[RSFWH98] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, 1998.

[SKS+08] Carlos Scheidegger, David Koop, Emanuele Santos, Huy Vo, Steven Callahan, Juliana Freire, and Claudio Silva. Tackling the provenance challenge one layer at a time. *Concurrency and Computation: Practice and Experience*, 20(5):473–483, 2008.

[SKV+07] Carlos Scheidegger, David Koop, Huy Vo, Juliana Freire, and Cláudio Silva. Querying and creating visualizations by analogy. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1560–1567, 2007.

[SLA+08] Emanuele Santos, Lauro Lins, James P. Ahrens, Juliana Freire, and Cláudio Silva. A first study on clustering collections of workflow graphs. In *Proceedings of the International Provenance and Annotation Workshop (IPAW)*, 2008.

[SPA08] SPARQL Query Language for RDF, `http://www.w3.org/TR/rdf-sparql-query/`, 2008.

[SPG05] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34(3):31–36, 2005.

[SSC09] Qihong Shao, Peng Sun, and Yi Chen. WISE: A Workflow Information Search Engine. Demo Description. In *ICDE*, pages 1491–1494, 2009.

[SVK+08] Carlos E. Scheidegger, Huy T. Vo, David Koop, Juliana Freire, and Claudio T. Silva. Querying and re-using workflows with vistrails. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1251–1254, 2008.

[Swi] The Swift System. `http://www.ci.uchicago.edu/swift`.

[Tav] The Taverna Project. `http://taverna.sourceforge.net`.

[Tri] The Triana Project. `http://www.trianacode.org`.

[TTHW07] A. Turpin, Y. Tsegay, D. Hawking, and H.E. Williams. Fast generation of result snippets in web search. In *SIGIR*, pages 127–134. ACM New York, NY, USA, 2007.

[VDS] VDS - The GriPhyN Virtual Data System. `http://www.ci.uchicago.edu/wiki/bin/view/VDS/VDSWeb/WebMain`.

[Vis] The VisTrails Project. `http://www.vistrails.org`.

[VR79] CJ Van Rijsbergen. *Information retrieval.* Butterworth-Heinemann Newton, MA, USA, 1979.

[VVS+08] Mark Vigder, Norman G. Vinson, Janice Singer, Darlene Stewart, and Keith Mews. Supporting scientists' everyday work: Automating scientific workflows. *IEEE Software*, 25(4):52–58, 2008.

[WBHW06] Haoxiang Wang, Ken Brodlie, James Handley, and Jason Wood. Service-oriented approach to collaborative visualization. In *Proceedings of UK e-Science All Hands Meeting 2006*, pages 241–248. National e-Science Centre, 2006.

[Wie92] Gio Wiederhold. Mediators in the architecture of future information systems. In Michael N. Huhns and Munindar P. Singh, editors, *Readings in Agents*, pages 185–196. Morgan Kaufmann, San Francisco, CA, USA, 1992.

[WKLW98] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. Dublin core metadata for resource discovery. *The Internet Society*, 1998.

[WWB97] Jason Wood, Helen Wright, and Ken Brodlie. Collaborative visualization. In *VIS '97: Proceedings of the 8th conference on Visualization '97*, pages 253–ff., Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[XM07] X. Xiang and G. Madey. Improving the reuse of scientific workflows and their by-products. In *Proceedings of 2007 IEEE International Conference on Web Services*, volume 00, pages 792–799, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[XML08] Extensible Markup Language (XML), `http://www.w3.org/XML/`, 2008.

[XPa08] XML Path Language (XPath), `http://www.w3.org/TR/xpath`, 2008.

[XQu08] XQuery 1.0: An XML Query Language, `http://www.w3.org/TR/xquery/`, 2008.

[Yah] Yahoo! Pipes. `http://pipes.yahoo.com`.

[YB05a] Jia Yu and Rajkumar Buyya. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Rec.*, 34(3):44–49, September 2005.

[YB05b] Jia Yu and Rajkumar Buyya. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Rec.*, 34(3):44–49, 2005.

[ZGS06] Jun Zhao, Carole Goble, and Robert Stevens. An identity crisis in the life sciences. In *Proc. of the 3rd International Provenance and Annotation Workshop*, Chicago, USA, May 2006. LNCS. extended paper.

[ZGST08] Jun Zhao, Carole Goble, Robert Stevens, and Daniele Turi. Mining taverna's semantic web of provenance. *Concurrency and Computation: Practice and Experience*, 20(5):463–472, 2008.

[ZHC$^+$04] Hua-Jun Zeng, Qi-Cai He, Zheng Chen, Wei-Ying Ma, and Jinwen Ma. Learning to cluster web search results. In *SIGIR*, pages 210–217, 2004.

[Zhu02] Hai Zhuge. A process matching approach for flexible workflow process reuse. *Information and Software Technology*, 44(8):445 – 450, 2002.

Avdelning, Institution Division, Department ADIT, Dept. of Computer and Information Science 581 83 Linköping	**Datum** Date 2010-01-15

Linköpings universitet

Språk Language ☐ Svenska/Swedish ☒ Engelska/English ☐ ______	**Rapporttyp** Report category ☒ Licentiatavhandling ☐ Examensarbete ☐ C-uppsats ☐ D-uppsats ☐ Övrig rapport ☐ ______	**ISBN** 978-91-7393-461-9 **ISRN** LiU-Tek-Lic–2009:35 **Serietitel och serienummer** Title of series, numbering **ISSN** 0280–7971 Linköping Studies in Science and Technology Thesis No. 1427

URL för elektronisk version

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-52017

Titel
Title

Supporting Scientific Collaboration through Workflows and Provenance

Författare
Author

Tommy Ellqvist

Sammanfattning
Abstract

Science is changing. Computers, high-speed communication, and new technologies have created new ways of conducting research. Researchers from different disciplines are processing and analyzing increasingly large volumes of scientific data. This kind of research requires that the scientists have access to tools that can handle large amounts of data, enable access to vast computational resources, and support the collaboration of large teams of scientists. This thesis focuses on tools that help support scientific collaboration.

Workflows and provenance have proven useful in supporting scientific collaboration. Workflows provide a formal specification of processes for scientific experiments, and provenance offers a model for documenting data and process dependencies. Together, they enable the creation of tools that can support collaboration through the life-cycle of scientific experiments, from specification of processes for scientific experiments to validation of results. However, existing models for workflows and provenance are often specific to particular tasks and tools. This makes it difficult to analyze the history of data that has been generated over several application areas by different tools. Moreover, designing workflows is time-consuming, often requires extensive knowledge of the tools involved, and may require collaboration with researchers with different expertise. This thesis addresses these problems.

Our first contribution is a study of the differences between two approaches to interoperability between provenance models: direct data conversion and mediation. We perform a case study where we integrate three different provenance models using the mediation approach, and show the advantages compared to data conversion. Our second contribution serves to support workflow design by allowing multiple users to concurrently design workflows. Current workflow tools do not allow users to work simultaneously on the same workflow. We propose a method that uses the provenance of workflow evolution to enable real-time collaborative design of workflows. Our third contribution considers supporting workflow design by reusing existing workflows. Workflow collections for reuse are available, but more efficient methods for generating summaries of search results are needed. We explore new summarization strategies that consider the workflow structure.

Nyckelord
Keywords

Scientific collaboration, Workflow, Provenance, Search Engine, Query Language, Data Integration

Department of Computer and Information Science
Linköpings universitet

Linköpings Studies in Science and Technology
Faculty of Arts and Sciences – Licentiate Theses

No 17	**Vojin Plavsic:** Interleaved Processing of Non-Numerical Data Stored on a Cyclic Memory. (Available at: FOA, Box 1165, S-581 11 Linköping, Sweden. FOA Report B30062E)
No 28	**Arne Jönsson, Mikael Patel:** An Interactive Flowcharting Technique for Communicating and Realizing Algorithms, 1984.
No 29	**Johnny Eckerland:** Retargeting of an Incremental Code Generator, 1984.
No 48	**Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching, 1985.
No 52	**Zebo Peng:** Steps Towards the Formalization of Designing VLSI Systems, 1985.
No 60	**Johan Fagerström:** Simulation and Evaluation of Architecture based on Asynchronous Processes, 1985.
No 71	**Jalal Maleki:** ICONStraint, A Dependency Directed Constraint Maintenance System, 1987.
No 72	**Tony Larsson:** On the Specification and Verification of VLSI Systems, 1986.
No 73	**Ola Strömfors:** A Structure Editor for Documents and Programs, 1986.
No 74	**Christos Levcopoulos:** New Results about the Approximation Behavior of the Greedy Triangulation, 1986.
No 104	**Shamsul I. Chowdhury:** Statistical Expert Systems - a Special Application Area for Knowledge-Based Computer Methodology, 1987.
No 108	**Rober Bilos:** Incremental Scanning and Token-Based Editing, 1987.
No 111	**Hans Block:** SPORT-SORT Sorting Algorithms and Sport Tournaments, 1987.
No 113	**Ralph Rönnquist:** Network and Lattice Based Approaches to the Representation of Knowledge, 1987.
No 118	**Mariam Kamkar, Nahid Shahmehri:** Affect-Chaining in Program Flow Analysis Applied to Queries of Programs, 1987.
No 126	**Dan Strömberg:** Transfer and Distribution of Application Programs, 1987.
No 127	**Kristian Sandahl:** Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems, 1987.
No 139	**Christer Bäckström:** Reasoning about Interdependent Actions, 1988.
No 140	**Mats Wirén:** On Control Strategies and Incrementality in Unification-Based Chart Parsing, 1988.
No 146	**Johan Hultman:** A Software System for Defining and Controlling Actions in a Mechanical System, 1988.
No 150	**Tim Hansen:** Diagnosing Faults using Knowledge about Malfunctioning Behavior, 1988.
No 165	**Jonas Löwgren:** Supporting Design and Management of Expert System User Interfaces, 1989.
No 166	**Ola Petersson:** On Adaptive Sorting in Sequential and Parallel Models, 1989.
No 174	**Yngve Larsson:** Dynamic Configuration in a Distributed Environment, 1989.
No 177	**Peter Åberg:** Design of a Multiple View Presentation and Interaction Manager, 1989.
No 181	**Henrik Eriksson:** A Study in Domain-Oriented Tool Support for Knowledge Acquisition, 1989.
No 184	**Ivan Rankin:** The Deep Generation of Text in Expert Critiquing Systems, 1989.
No 187	**Simin Nadjm-Tehrani:** Contributions to the Declarative Approach to Debugging Prolog Programs, 1989.
No 189	**Magnus Merkel:** Temporal Information in Natural Language, 1989.
No 196	**Ulf Nilsson:** A Systematic Approach to Abstract Interpretation of Logic Programs, 1989.
No 197	**Staffan Bonnier:** Horn Clause Logic with External Procedures: Towards a Theoretical Framework, 1989.
No 203	**Christer Hansson:** A Prototype System for Logical Reasoning about Time and Action, 1990.
No 212	**Björn Fjellborg:** An Approach to Extraction of Pipeline Structures for VLSI High-Level Synthesis, 1990.
No 230	**Patrick Doherty:** A Three-Valued Approach to Non-Monotonic Reasoning, 1990.
No 237	**Tomas Sokolnicki:** Coaching Partial Plans: An Approach to Knowledge-Based Tutoring, 1990.
No 250	**Lars Strömberg:** Postmortem Debugging of Distributed Systems, 1990.
No 253	**Torbjörn Näslund:** SLDFA-Resolution - Computing Answers for Negative Queries, 1990.
No 260	**Peter D. Holmes:** Using Connectivity Graphs to Support Map-Related Reasoning, 1991.
No 283	**Olof Johansson:** Improving Implementation of Graphical User Interfaces for Object-Oriented Knowledge- Bases, 1991.
No 298	**Rolf G Larsson:** Aktivitetsbaserad kalkylering i ett nytt ekonomisystem, 1991.
No 318	**Lena Srömbäck:** Studies in Extended Unification-Based Formalism for Linguistic Description: An Algorithm for Feature Structures with Disjunction and a Proposal for Flexible Systems, 1992.
No 319	**Mikael Pettersson:** DML-A Language and System for the Generation of Efficient Compilers from Denotational Specification, 1992.
No 326	**Andreas Kågedal:** Logic Programming with External Procedures: an Implementation, 1992.
No 328	**Patrick Lambrix:** Aspects of Version Management of Composite Objects, 1992.
No 333	**Xinli Gu:** Testability Analysis and Improvement in High-Level Synthesis Systems, 1992.
No 335	**Torbjörn Näslund:** On the Role of Evaluations in Iterative Development of Managerial Support Systems, 1992.
No 348	**Ulf Cederling:** Industrial Software Development - a Case Study, 1992.
No 352	**Magnus Morin:** Predictable Cyclic Computations in Autonomous Systems: A Computational Model and Implementation, 1992.
No 371	**Mehran Noghabai:** Evaluation of Strategic Investments in Information Technology, 1993.
No 378	**Mats Larsson:** A Transformational Approach to Formal Digital System Design, 1993.
No 380	**Johan Ringström:** Compiler Generation for Parallel Languages from Denotational Specifications, 1993.

No 381 **Michael Jansson:** Propagation of Change in an Intelligent Information System, 1993.
No 383 **Jonni Harrius:** An Architecture and a Knowledge Representation Model for Expert Critiquing Systems, 1993.
No 386 **Per Österling:** Symbolic Modelling of the Dynamic Environments of Autonomous Agents, 1993.
No 398 **Johan Boye:** Dependency-based Groudness Analysis of Functional Logic Programs, 1993.
No 402 **Lars Degerstedt:** Tabulated Resolution for Well Founded Semantics, 1993.
No 406 **Anna Moberg:** Satellitkontor - en studie av kommunikationsmönster vid arbete på distans, 1993.
No 414 **Peter Carlsson:** Separation av företagsledning och finansiering - fallstudier av företagsledarutköp ur ett agent-teoretiskt perspektiv, 1994.
No 417 **Camilla Sjöström:** Revision och lagreglering - ett historiskt perspektiv, 1994.
No 436 **Cecilia Sjöberg:** Voices in Design: Argumentation in Participatory Development, 1994.
No 437 **Lars Viklund:** Contributions to a High-level Programming Environment for a Scientific Computing, 1994.
No 440 **Peter Loborg:** Error Recovery Support in Manufacturing Control Systems, 1994.
FHS 3/94 **Owen Eriksson:** Informationssystem med verksamhetskvalitet - utvärdering baserat på ett verksamhetsinriktat och samskapande perspektiv, 1994.
FHS 4/94 **Karin Pettersson:** Informationssystemstrukturering, ansvarsfördelning och användarinflytande - En komparativ studie med utgångspunkt i två informationssystemstrategier, 1994.
No 441 **Lars Poignant:** Informationsteknologi och företagsetablering - Effekter på produktivitet och region, 1994.
No 446 **Gustav Fahl:** Object Views of Relational Data in Multidatabase Systems, 1994.
No 450 **Henrik Nilsson:** A Declarative Approach to Debugging for Lazy Functional Languages, 1994.
No 451 **Jonas Lind:** Creditor - Firm Relations: an Interdisciplinary Analysis, 1994.
No 452 **Martin Sköld:** Active Rules based on Object Relational Queries - Efficient Change Monitoring Techniques, 1994.
No 455 **Pär Carlshamre:** A Collaborative Approach to Usability Engineering: Technical Communicators and System Developers in Usability-Oriented Systems Development, 1994.
FHS 5/94 **Stefan Cronholm:** Varför CASE-verktyg i systemutveckling? - En motiv- och konsekvensstudie avseende arbetssätt och arbetsformer, 1994.
No 462 **Mikael Lindvall:** A Study of Traceability in Object-Oriented Systems Development, 1994.
No 463 **Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av Sandviks förvärv av Bahco Verktyg, 1994.
No 464 **Hans Olsén:** Collage Induction: Proving Properties of Logic Programs by Program Synthesis, 1994.
No 469 **Lars Karlsson:** Specification and Synthesis of Plans Using the Features and Fluents Framework, 1995.
No 473 **Ulf Söderman:** On Conceptual Modelling of Mode Switching Systems, 1995.
No 475 **Choong-ho Yi:** Reasoning about Concurrent Actions in the Trajectory Semantics, 1995.
No 476 **Bo Lagerström:** Successiv resultatavräkning av pågående arbeten. - Fallstudier i tre byggföretag, 1995.
No 478 **Peter Jonsson:** Complexity of State-Variable Planning under Structural Restrictions, 1995.
FHS 7/95 **Anders Avdic:** Arbetsintegrerad systemutveckling med kalkylprogram, 1995.
No 482 **Eva L Ragnemalm:** Towards Student Modelling through Collaborative Dialogue with a Learning Companion, 1995.
No 488 **Eva Toller:** Contributions to Parallel Multiparadigm Languages: Combining Object-Oriented and Rule-Based Programming, 1995.
No 489 **Erik Stoy:** A Petri Net Based Unified Representation for Hardware/Software Co-Design, 1995.
No 497 **Johan Herber:** Environment Support for Building Structured Mathematical Models, 1995.
No 498 **Stefan Svenberg:** Structure-Driven Derivation of Inter-Lingual Functor-Argument Trees for Multi-Lingual Generation, 1995.
No 503 **Hee-Cheol Kim:** Prediction and Postdiction under Uncertainty, 1995.
FHS 8/95 **Dan Fristedt:** Metoder i användning - mot förbättring av systemutveckling genom situationell metodkunskap och metodanalys, 1995.
FHS 9/95 **Malin Bergvall:** Systemförvaltning i praktiken - en kvalitativ studie avseende centrala begrepp, aktiviteter och ansvarsroller, 1995.
No 513 **Joachim Karlsson:** Towards a Strategy for Software Requirements Selection, 1995.
No 517 **Jakob Axelsson:** Schedulability-Driven Partitioning of Heterogeneous Real-Time Systems, 1995.
No 518 **Göran Forslund:** Toward Cooperative Advice-Giving Systems: The Expert Systems Experience, 1995.
No 522 **Jörgen Andersson:** Bilder av småföretagares ekonomistyrning, 1995.
No 538 **Staffan Flodin:** Efficient Management of Object-Oriented Queries with Late Binding, 1996.
No 545 **Vadim Engelson:** An Approach to Automatic Construction of Graphical User Interfaces for Applications in Scientific Computing, 1996.
No 546 **Magnus Werner :** Multidatabase Integration using Polymorphic Queries and Views, 1996.
FiF-a 1/96 **Mikael Lind:** Affärsprocessinriktad förändringsanalys - utveckling och tillämpning av synsätt och metod, 1996.
No 549 **Jonas Hallberg:** High-Level Synthesis under Local Timing Constraints, 1996.
No 550 **Kristina Larsen:** Förutsättningar och begränsningar för arbete på distans - erfarenheter från fyra svenska företag. 1996.
No 557 **Mikael Johansson:** Quality Functions for Requirements Engineering Methods, 1996.
No 558 **Patrik Nordling:** The Simulation of Rolling Bearing Dynamics on Parallel Computers, 1996.
No 561 **Anders Ekman:** Exploration of Polygonal Environments, 1996.
No 563 **Niclas Andersson:** Compilation of Mathematical Models to Parallel Code, 1996.
No 567 **Johan Jenvald:** Simulation and Data Collection in Battle Training, 1996.

No 575	**Niclas Ohlsson:** Software Quality Engineering by Early Identification of Fault-Prone Modules, 1996.
No 576	**Mikael Ericsson:** Commenting Systems as Design Support—A Wizard-of-Oz Study, 1996.
No 587	**Jörgen Lindström:** Chefers användning av kommunikationsteknik, 1996.
No 589	**Esa Falkenroth:** Data Management in Control Applications - A Proposal Based on Active Database Systems, 1996.
No 591	**Niclas Wahllöf:** A Default Extension to Description Logics and its Applications, 1996.
No 595	**Annika Larsson:** Ekonomisk Styrning och Organisatorisk Passion - ett interaktivt perspektiv, 1997.
No 597	**Ling Lin:** A Value-based Indexing Technique for Time Sequences, 1997.
No 598	**Rego Granlund:** C^3Fire - A Microworld Supporting Emergency Management Training, 1997.
No 599	**Peter Ingels:** A Robust Text Processing Technique Applied to Lexical Error Recovery, 1997.
No 607	**Per-Arne Persson:** Toward a Grounded Theory for Support of Command and Control in Military Coalitions, 1997.
No 609	**Jonas S Karlsson:** A Scalable Data Structure for a Parallel Data Server, 1997.
FiF-a 4	**Carita Åbom:** Videomötesteknik i olika affärssituationer - möjligheter och hinder, 1997.
FiF-a 6	**Tommy Wedlund**: Att skapa en företagsanpassad systemutvecklingsmodell - genom rekonstruktion, värdering och vidareutveckling i T50-bolag inom ABB, 1997.
No 615	**Silvia Coradeschi**: A Decision-Mechanism for Reactive and Coordinated Agents, 1997.
No 623	**Jan Ollinen:** Det flexibla kontorets utveckling på Digital - Ett stöd för multiflex? 1997.
No 626	**David Byers:** Towards Estimating Software Testability Using Static Analysis, 1997.
No 627	**Fredrik Eklund:** Declarative Error Diagnosis of GAPLog Programs, 1997.
No 629	**Gunilla Ivefors:** Krigsspel och Informationsteknik inför en oförutsägbar framtid, 1997**.**
No 631	**Jens-Olof Lindh:** Analysing Traffic Safety from a Case-Based Reasoning Perspective, 1997
No 639	**Jukka Mäki-Turja:**. Smalltalk - a suitable Real-Time Language, 1997.
No 640	**Juha Takkinen**: CAFE: Towards a Conceptual Model for Information Management in Electronic Mail, 1997.
No 643	**Man Lin**: Formal Analysis of Reactive Rule-based Programs, 1997.
No 653	**Mats Gustafsson**: Bringing Role-Based Access Control to Distributed Systems, 1997.
FiF-a 13	**Boris Karlsson:** Metodanalys för förståelse och utveckling av systemutvecklingsverksamhet. Analys och värdering av systemutvecklingsmodeller och dess användning, 1997.
No 674	**Marcus Bjäreland:** Two Aspects of Automating Logics of Action and Change - Regression and Tractability, 1998.
No 676	**Jan Håkegård**: Hierarchical Test Architecture and Board-Level Test Controller Synthesis, 1998.
No 668	**Per-Ove Zetterlund**: Normering av svensk redovisning - En studie av tillkomsten av Redovisningsrådets rekommendation om koncernredovisning (RR01:91), 1998.
No 675	**Jimmy Tjäder**: Projektledaren & planen - en studie av projektledning i tre installations- och systemutvecklingsprojekt, 1998.
FiF-a 14	**Ulf Melin**: Informationssystem vid ökad affärs- och processorientering - egenskaper, strategier och utveckling, 1998.
No 695	**Tim Heyer**: COMPASS: Introduction of Formal Methods in Code Development and Inspection, 1998.
No 700	**Patrik Hägglund:** Programming Languages for Computer Algebra, 1998.
FiF-a 16	**Marie-Therese Christiansson:** Inter-organisatorisk verksamhetsutveckling - metoder som stöd vid utveckling av partnerskap och informationssystem, 1998.
No 712	**Christina Wennestam:** Information om immateriella resurser. Investeringar i forskning och utveckling samt i personal inom skogsindustrin, 1998.
No 719	**Joakim Gustafsson:** Extending Temporal Action Logic for Ramification and Concurrency, 1998.
No 723	**Henrik André-Jönsson:** Indexing time-series data using text indexing methods, 1999.
No 725	**Erik Larsson:** High-Level Testability Analysis and Enhancement Techniques, 1998.
No 730	**Carl-Johan Westin:** Informationsförsörjning: en fråga om ansvar - aktiviteter och uppdrag i fem stora svenska organisationers operativa informationsförsörjning, 1998.
No 731	**Åse Jansson:** Miljöhänsyn - en del i företags styrning, 1998.
No 733	**Thomas Padron-McCarthy:** Performance-Polymorphic Declarative Queries, 1998.
No 734	**Anders Bäckström:** Värdeskapande kreditgivning - Kreditriskhantering ur ett agentteoretiskt perspektiv, 1998.
FiF-a 21	**Ulf Seigerroth:** Integration av förändringsmetoder - en modell för välgrundad metodintegration, 1999.
FiF-a 22	**Fredrik Öberg:** Object-Oriented Frameworks - A New Strategy for Case Tool Development, 1998.
No 737	**Jonas Mellin:** Predictable Event Monitoring, 1998.
No 738	**Joakim Eriksson:** Specifying and Managing Rules in an Active Real-Time Database System, 1998.
FiF-a 25	**Bengt E W Andersson:** Samverkande informationssystem mellan aktörer i offentliga åtaganden - En teori om aktörsarenor i samverkan om utbyte av information, 1998.
No 742	**Pawel Pietrzak:** Static Incorrectness Diagnosis of CLP (FD), 1999.
No 748	**Tobias Ritzau:** Real-Time Reference Counting in RT-Java, 1999.
No 751	**Anders Ferntoft:** Elektronisk affärskommunikation - kontaktkostnader och kontaktprocesser mellan kunder och leverantörer på producentmarknader, 1999.
No 752	**Jo Skåmedal:** Arbete på distans och arbetsformens påverkan på resor och resmönster, 1999.
No 753	**Johan Alvehus:** Mötets metaforer. En studie av berättelser om möten, 1999.
No 754	**Magnus Lindahl:** Bankens villkor i låneavtal vid kreditgivning till högt belånade företagsförvärv: En studie ur ett agentteoretiskt perspektiv, 2000.

No 766	**Martin V. Howard:** Designing dynamic visualizations of temporal data, 1999.
No 769	**Jesper Andersson:** Towards Reactive Software Architectures, 1999.
No 775	**Anders Henriksson:** Unique kernel diagnosis, 1999.
FiF-a 30	**Pär J. Ågerfalk:** Pragmatization of Information Systems - A Theoretical and Methodological Outline, 1999.
No 787	**Charlotte Björkegren:** Learning for the next project - Bearers and barriers in knowledge transfer within an organisation, 1999.
No 788	**Håkan Nilsson:** Informationsteknik som drivkraft i granskningsprocessen - En studie av fyra revisionsbyråer, 2000.
No 790	**Erik Berglund:** Use-Oriented Documentation in Software Development, 1999.
No 791	**Klas Gäre:** Verksamhetsförändringar i samband med IS-införande, 1999.
No 800	**Anders Subotic:** Software Quality Inspection, 1999.
No 807	**Svein Bergum**: Managerial communication in telework, 2000.
No 809	**Flavius Gruian:** Energy-Aware Design of Digital Systems, 2000.
FiF-a 32	**Karin Hedström:** Kunskapsanvändning och kunskapsutveckling hos verksamhetskonsulter - Erfarenheter från ett FOU-samarbete, 2000.
No 808	**Linda Askenäs:** Affärssystemet - En studie om teknikens aktiva och passiva roll i en organisation, 2000.
No 820	**Jean Paul Meynard:** Control of industrial robots through high-level task programming, 2000.
No 823	**Lars Hult:** Publika Gränsytor - ett designexempel, 2000.
No 832	**Paul Pop:** Scheduling and Communication Synthesis for Distributed Real-Time Systems, 2000.
FiF-a 34	**Göran Hultgren:** Nätverksinriktad Förändringsanalys - perspektiv och metoder som stöd för förståelse och utveckling av affärsrelationer och informationssystem, 2000.
No 842	**Magnus Kald:** The role of management control systems in strategic business units, 2000.
No 844	**Mikael Cäker:** Vad kostar kunden? Modeller för intern redovisning, 2000.
FiF-a 37	**Ewa Braf**: Organisationers kunskapsverksamheter - en kritisk studie av ”knowledge management”, 2000.
FiF-a 40	**Henrik Lindberg:** Webbaserade affärsprocesser - Möjligheter och begränsningar, 2000.
FiF-a 41	**Benneth Christiansson:** Att komponentbasera informationssystem - Vad säger teori och praktik?, 2000.
No. 854	**Ola Pettersson:** Deliberation in a Mobile Robot, 2000.
No 863	**Dan Lawesson**: Towards Behavioral Model Fault Isolation for Object Oriented Control Systems, 2000.
No 881	**Johan Moe:** Execution Tracing of Large Distributed Systems, 2001.
No 882	**Yuxiao Zhao:** XML-based Frameworks for Internet Commerce and an Implementation of B2B e-procurement, 2001.
No 890	**Annika Flycht-Eriksson:** Domain Knowledge Management in Information-providing Dialogue systems, 2001.
FiF-a 47	**Per-Arne Segerkvis**t: Webbaserade imaginära organisationers samverkansformer: Informationssystemarkitektur och aktörssamverkan som förutsättningar för affärsprocesser, 2001.
No 894	**Stefan Svarén:** Styrning av investeringar i divisionaliserade företag - Ett koncernperspektiv, 2001.
No 906	**Lin Han**: Secure and Scalable E-Service Software Delivery, 2001.
No 917	**Emma Hansson:** Optionsprogram för anställda - en studie av svenska börsföretag, 2001.
No 916	**Susanne Odar:** IT som stöd för strategiska beslut, en studie av datorimplementerade modeller av verksamhet som stöd för beslut om anskaffning av JAS 1982, 2002.
FiF-a-49	**Stefan Holgersson:** IT-system och filtrering av verksamhetskunskap - kvalitetsproblem vid analyser och beslutsfattande som bygger på uppgifter hämtade från polisens IT-system, 2001.
FiF-a-51	**Per Oscarsson:** Informationssäkerhet i verksamheter - begrepp och modeller som stöd för förståelse av informationssäkerhet och dess hantering, 2001.
No 919	**Luis Alejandro Cortes:** A Petri Net Based Modeling and Verification Technique for Real-Time Embedded Systems, 2001.
No 915	**Niklas Sandell:** Redovisning i skuggan av en bankkris - Värdering av fastigheter. 2001.
No 931	**Fredrik Elg:** Ett dynamiskt perspektiv på individuella skillnader av heuristisk kompetens, intelligens, mentala modeller, mål och konfidens i kontroll av mikrovärlden Moro, 2002.
No 933	**Peter Aronsson:** Automatic Parallelization of Simulation Code from Equation Based Simulation Languages, 2002.
No 938	**Bourhane Kadmiry**: Fuzzy Control of Unmanned Helicopter, 2002.
No 942	**Patrik Haslum**: Prediction as a Knowledge Representation Problem: A Case Study in Model Design, 2002.
No 956	**Robert Sevenius:** On the instruments of governance - A law & economics study of capital instruments in limited liability companies, 2002.
FiF-a 58	**Johan Petersson:** Lokala elektroniska marknadsplatser - informationssystem för platsbundna affärer, 2002.
No 964	**Peter Bunus:** Debugging and Structural Analysis of Declarative Equation-Based Languages, 2002.
No 973	**Gert Jervan:** High-Level Test Generation and Built-In Self-Test Techniques for Digital Systems, 2002.
No 958	**Fredrika Berglund:** Management Control and Strategy - a Case Study of Pharmaceutical Drug Development, 2002.
FiF-a 61	**Fredrik Karlsson:** Meta-Method for Method Configuration - A Rational Unified Process Case, 2002.
No 985	**Sorin Manolache:** Schedulability Analysis of Real-Time Systems with Stochastic Task Execution Times, 2002.
No 982	**Diana Szentiványi:** Performance and Availability Trade-offs in Fault-Tolerant Middleware, 2002.
No 989	**Iakov Nakhimovski:** Modeling and Simulation of Contacting Flexible Bodies in Multibody Systems, 2002.
No 990	**Levon Saldamli:** PDEModelica - Towards a High-Level Language for Modeling with Partial Differential Equations, 2002.
No 991	**Almut Herzog:** Secure Execution Environment for Java Electronic Services, 2002.
No 999	**Jon Edvardsson:** Contributions to Program- and Specification-based Test Data Generation, 2002.
No 1000	**Anders Arpteg:** Adaptive Semi-structured Information Extraction, 2002.

No 1001	**Andrzej Bednarski:** A Dynamic Programming Approach to Optimal Retargetable Code Generation for Irregular Architectures, 2002.
No 988	**Mattias Arvola:** Good to use! : Use quality of multi-user applications in the home, 2003.
FiF-a 62	**Lennart Ljung:** Utveckling av en projektivitetsmodell - om organisationers förmåga att tillämpa projektarbetsformen, 2003.
No 1003	**Pernilla Qvarfordt:** User experience of spoken feedback in multimodal interaction, 2003.
No 1005	**Alexander Siemers:** Visualization of Dynamic Multibody Simulation With Special Reference to Contacts, 2003.
No 1008	**Jens Gustavsson:** Towards Unanticipated Runtime Software Evolution, 2003.
No 1010	**Calin Curescu:** Adaptive QoS-aware Resource Allocation for Wireless Networks, 2003.
No 1015	**Anna Andersson:** Management Information Systems in Process-oriented Healthcare Organisations, 2003.
No 1018	**Björn Johansson:** Feedforward Control in Dynamic Situations, 2003.
No 1022	**Traian Pop:** Scheduling and Optimisation of Heterogeneous Time/Event-Triggered Distributed Embedded Systems, 2003.
FiF-a 65	**Britt-Marie Johansson:** Kundkommunikation på distans - en studie om kommunikationsmediets betydelse i affärstransaktioner, 2003.
No 1024	**Aleksandra Tešanovic:** Towards Aspectual Component-Based Real-Time System Development, 2003.
No 1034	**Arja Vainio-Larsson**: Designing for Use in a Future Context - Five Case Studies in Retrospect, 2003.
No 1033	**Peter Nilsson:** Svenska bankers redovisningsval vid reservering för befarade kreditförluster - En studie vid införandet av nya redovisningsregler, 2003.
FiF-a 69	**Fredrik Ericsson:** Information Technology for Learning and Acquiring of Work Knowledge, 2003.
No 1049	**Marcus Comstedt:** Towards Fine-Grained Binary Composition through Link Time Weaving, 2003.
No 1052	**Åsa Hedenskog:** Increasing the Automation of Radio Network Control, 2003.
No 1054	**Claudiu Duma:** Security and Efficiency Tradeoffs in Multicast Group Key Management, 2003.
FiF-a 71	**Emma Eliason:** Effektanalys av IT-systems handlingsutrymme, 2003.
No 1055	**Carl Cederberg:** Experiments in Indirect Fault Injection with Open Source and Industrial Software, 2003.
No 1058	**Daniel Karlsson:** Towards Formal Verification in a Component-based Reuse Methodology, 2003.
FiF-a 73	**Anders Hjalmarsson:** Att etablera och vidmakthålla förbättringsverksamhet - behovet av koordination och interaktion vid förändring av systemutvecklingsverksamheter, 2004.
No 1079	**Pontus Johansson:** Design and Development of Recommender Dialogue Systems, 2004.
No 1084	**Charlotte Stoltz:** Calling for Call Centres - A Study of Call Centre Locations in a Swedish Rural Region, 2004.
FiF-a 74	**Björn Johansson:** Deciding on Using Application Service Provision in SMEs, 2004.
No 1094	**Genevieve Gorrell:** Language Modelling and Error Handling in Spoken Dialogue Systems, 2004.
No 1095	**Ulf Johansson:** Rule Extraction - the Key to Accurate and Comprehensible Data Mining Models, 2004.
No 1099	**Sonia Sangari:** Computational Models of Some Communicative Head Movements, 2004.
No 1110	**Hans Nässla:** Intra-Family Information Flow and Prospects for Communication Systems, 2004.
No 1116	**Henrik Sällberg:** On the value of customer loyalty programs - A study of point programs and switching costs, 2004.
FiF-a 77	**Ulf Larsson:** Designarbete i dialog - karaktärisering av interaktionen mellan användare och utvecklare i en systemutvecklingsprocess, 2004.
No 1126	**Andreas Borg:** Contribution to Management and Validation of Non-Functional Requirements, 2004.
No 1127	**Per-Ola Kristensson:** Large Vocabulary Shorthand Writing on Stylus Keyboard, 2004.
No 1132	**Pär-Anders Albinsson:** Interacting with Command and Control Systems: Tools for Operators and Designers, 2004.
No 1130	**Ioan Chisalita:** Safety-Oriented Communication in Mobile Networks for Vehicles, 2004.
No 1138	**Thomas Gustafsson:** Maintaining Data Consistency in Embedded Databases for Vehicular Systems, 2004.
No 1149	**Vaida Jakoniené:** A Study in Integrating Multiple Biological Data Sources, 2005.
No 1156	**Abdil Rashid Mohamed**: High-Level Techniques for Built-In Self-Test Resources Optimization, 2005.
No 1162	**Adrian Pop:** Contributions to Meta-Modeling Tools and Methods, 2005.
No 1165	**Fidel Vascós Palacios:** On the information exchange between physicians and social insurance officers in the sick leave process: an Activity Theoretical perspective, 2005.
FiF-a 84	**Jenny Lagsten:** Verksamhetsutvecklande utvärdering i informationssystemprojekt, 2005.
No 1166	**Emma Larsdotter Nilsson:** Modeling, Simulation, and Visualization of Metabolic Pathways Using Modelica, 2005.
No 1167	**Christina Keller:** Virtual Learning Environments in higher education. A study of students' acceptance of educational technology, 2005.
No 1168	**Cécile Åberg:** Integration of organizational workflows and the Semantic Web, 2005.
FiF-a 85	**Anders Forsman:** Standardisering som grund för informationssamverkan och IT-tjänster - En fallstudie baserad på trafikinformationstjänsten RDS-TMC, 2005.
No 1171	**Yu-Hsing Huang:** A systemic traffic accident model, 2005.
FiF-a 86	**Jan Olausson:** Att modellera uppdrag - grunder för förståelse av processinriktade informationssystem i transaktionsintensiva verksamheter, 2005.
No 1172	**Petter Ahlström**: Affärsstrategier för seniorbostadsmarknaden, 2005.
No 1183	**Mathias Cöster**: Beyond IT and Productivity - How Digitization Transformed the Graphic Industry, 2005.
No 1184	**Åsa Horzella**: Beyond IT and Productivity - Effects of Digitized Information Flows in Grocery Distribution, 2005.
No 1185	**Maria Kollberg**: Beyond IT and Productivity - Effects of Digitized Information Flows in the Logging Industry, 2005.
No 1190	**David Dinka**: Role and Identity - Experience of technology in professional settings, 2005.
No 1191	**Andreas Hansson**: Increasing the Storage Capacity of Recursive Auto-associative Memory by Segmenting Data, 2005.