Search, Adapt, and Reuse: The Future of Scientific Workflows

Sarah Cohen-Boulakia
Université Paris-Sud 11 CNRS
UMR 8623, AMIB INRIA Saclay
cohen@lri.fr

Ulf Leser
Humboldt-Universität zu Berlin
Unter den Linden 6, 10099 Berlin
leser@informatik.hu-berlin.de

ABSTRACT

Over the last years, a number of scientific workflow management systems (SciWFM) have been brought to a state of maturity that should permit their usage in a production-style environment. This is especially true for the Life Sciences, but SciWFM also attract considerable attention in fields like geophysics or climate research. These developments, accompanied by the growing availability of analytical tools wrapped as (web) services, were driven by a series of very interesting promises: End users will be empowered to develop their own pipelines; reuse of services will be enhanced by easier integration into custom workflows; time necessary for developing analysis pipelines will decrease; etc. But despite all efforts, SciWFM have not yet found widespread acceptance in their intended audience. In this paper, we argue that a wider adoption of SciWFM will only be achieved if the focus of research and development is shifted from methods for developing and running workflows to searching, adapting, and reusing existing workflows. Only by this shift can SciWFM outreach to the mass of domain scientists actually performing scientific analysis – and with little interest in developing them themselves. To this end, SciWFM need to be combined with community-wide workflow repositories allowing users to find solutions for their scientific needs (coded as a workflow). In this vision paper, we show how and where such developments have already started and highlight new research questions arising.

Keywords

Scientific Workflow Systems, Workflow Management, Scientific data, Data Analysis.

1. INTRODUCTION

In the last decade, considerable effort has been put into the development of scientific workflow management systems. These systems primarily aim at supporting domain scientists in developing, running, and monitoring data analysis programs coded as workflows [DGST09]. A variety of systems, including Taverna [OGA+05], Kepler [LAB+05], Pegasus [DSS+04], VisTrails [SVK+08], and Triana [CGH+06], have reached a level of maturity that in principle allows them to be used by scientists for their daily needs. The goal of these systems is to put away the disadvantages of the state-of-the-art in developing scientific analysis, which is implementing custom programs, mostly in Perl or similar scripting languages. SciWFM promise (to varying degrees) to enable development of analysis pipelines at a higher level of abstraction, to take care of logging, provenance management, process control, recovery, scheduling and parallelization of individual tasks, and to increase understandability and sharing of workflows. SciWFM also could be a key infrastructure for repeatable science [GNT+10].

Until now, however, uptake of SciWFM has been limited. A search for "scientific workflow management" in Google Scholar lists more than 1,000 papers. However, many of them are concerned with research in SciWFM involving Computer Scientists, and not with their use by scientists. For instance in the Life Sciences, one of the major focuses of papers on SciWFM, we are aware of only a few papers that report on using a SciWFM for science, many of which were co-authored by developers of SciWFM systems. Though it might be that more users use SciWFM without mentioning it in their publications, we believe that it is safe to state that SciWFM have not yet reached their intended user group to a satisfying level.

There are many speculations on reasons for this situation. We believe that, actually, SciWFM have yet failed to properly define and target their intended user group. For a domain scientist, today's systems are much too complex. In the same manner as such users don't want to write Perl scripts or SQL queries, they don't want to model and program analysis workflows. Furthermore, for these people a workflow is not easier to understand than a program; on the other hand, these are exactly the people that have regular and pressing needs for performing data analysis on the masses of new data they generate daily [AKD10]. The competitors for SciWFM with respect to this user group are packed (and potentially commercial) applications (in Transcriptomics, from which we will draw our examples in the rest of this papers, this would be systems like Chipster or GeneSpring). For a Bioinformatician, on the other hand, SciWFM are too cumbersome and inflexible. These users are mostly interested in developing new methods and are usually capable of writing their own analysis scripts [HMB07]. Today they can take advantage of the large number of freely available libraries (such as BioPerl, BioJava, BioSQL, or R) for performing standard tasks, which already let them concentrate much more on their problem at hand than it was possible 10 years ago. A SciWFM offers them few advantages over scripting.

Furthermore, SciWFM have some inherent properties diminishing their potential advantages. First, a complex workflow composed of dozens of intertwined tasks, in general, is not much easier to understand than a well structured program performing the same analysis. Second, the ability to seamlessly integrate external services often leads to bad performance (as large amounts of data need to be passed around in wide-area networks), turns debugging into a nightmare, and also makes workflow execution dependent on the reliability of these services. Finally, experiences show that services are easily listed in a repository, but that they do not easily work together due to incompatibilities of data types, formats, invocation methods etc. Instead of concentrating on the scientific tasks, developers of scientific workflows therefore still have to spend a big deal of their time in writing "glue" code [RLS+06].

These observations call for a fresh view on SciWFM. We argue that SciWFM will have a hard time in persuading developers of new analysis methods to use them (called "power users" from now on). At the same time, by focusing exactly on this class of people, the SciWFM community ignores the much larger group of users that actually could benefit the most of SciWFM, i.e., the domain scientists ("true users" from now on) [Ste08]. These people do not know how to program, and they don't know how to use a SciWFM to develop a workflow – and they probably never will. However, true users are in urgent need for new methods to analyze their ever growing flood of data. They are interested in applying advanced methods (which they don't even need to understand) on new data to discover new facts in their respective science. Essentially, they are searching a solution to a given problem instead of aiming at developing a solution themselves. This currently bounds them to packaged software, often commercial, which ties them to the vendor / creator and cuts them off from using the most recent methods. At the same time, they would benefit tremendously from all the runtime-support that a SciWFM offers – but the technical obstacles currently are too severe for them to be able to take advantage.

In this paper, we provide a vision on what we believe could be the killer application for SciWFM – focusing on true users. Essentially, we take a new point-of-view on an existing class of systems which would make them attractive for masses of new users. We describe use cases that would be possible when following our novel point-of-view, show existing developments in this direction, and describe technical challenges arising from our vision. Adequately supporting true users requires a couple of things, such as domain-specific repositories of existing workflows, methods to find workflows solving a given analysis problem, and algorithms for comparing different workflows solving the same problem. Taken to the extreme, in our vision SciWFM will more and more become a largely invisible part of the computational infrastructure of a scientist, such as a database or an operating system is, that most (true) users do not directly deal with anymore. Instead, they simply chose a workflow, provide their data, maybe set some parameters, and press the "run" button. All the technology for scheduling, monitoring, restarting, service invocation, format conversion etc. remains hidden.

Notably, we do not believe that power users should or will stop using SciWFM. These people, though not in numbers, will continue to use SciWFM for developing new methods. Actually, if the true users would start to use SciWFM more often, this could also increase the attractiveness of SciWFM for power users, as publishing their workflows in such a system would increase the chances for them of being reused.

2. State-of-the-Art in SciWFM

This section gives an overview over the state-of-the-art of current SciWFM systems. We focus on those concepts that are critical for the support of "true users". For a general review on SciWFM see [DGST09].

2.1 Workflow Design

One of the most important claims of SciWFM it to provide an environment to ease the design of scientific analysis processes. The claim is that by representing these processes at a higher level of abstraction, their understandability, reuse and modularity would be enhanced [LAB+05]. Along this line, all SciWFM are equipped with a GUI in which users may drag and drop boxes (tasks) and connect them to describe data or control flow which should raise the general level of feeling "it is simple".

However, while domain scientists usually have an idea of the kind of analysis they want to perform on their particular data set, they often have no detailed knowledge of which concrete tools to use (and the parameters to set) and how to combine them into a complete workflow that would execute their intention. Several proposals have appeared to help in this respect. First, repositories of tools and services have emerged, such as BioCatalogue listing more than 3,000 web services [BTN+10] indexed by keywords and ontology terms. Second, tools have been developed that allow automatic translation of abstract specifications into concrete workflows [GGW+09; LAG03]. However, we have doubts whether such attempts are particularly helpful for the true user. Large repositories are impressive, but searching them is hard. True users are rather overstrained than supported by the amount of services being offered [DRGS09]. They are often not capable of (or not willing to) taking a rational choice. Furthermore, users still have the major problem of concretely chaining the tools together to solve their problem. On the other hand, systems supporting workflow inference require a level of detail in description of the available services that may render them inapplicable in any larger scenario.

2.2 Models of Workflows and Workflow Runs

In SciWFM, one has to distinguish between a workflow specification, given in some kind of modeling language, and the concrete workflow runs (or traces), which are series of executed tasks that conform to a workflow specification. Both aspects are treated very differently in different systems.

1. Currently, essentially every SciWFM has their particular language for specifying a workflow, equipped with a particular semantics of how such a specification is interpreted as an executable program [MLB+10]. Things get even more complicated with systems that allow dynamic adaptation of the semantics of a specification; for instance, the Kepler system offers different so-called directors for orchestrating workflow execution, leading to entirely different execution threads [DKM+05].
2. Storing, modeling and searching runs recently have become prominent research topics, usually summarized under the term "provenance management". Provenance is a key concept for SciWFM since it supports reproducibility of how results were obtained and helps assessing their quality [BCB+08; DF08]. Approaches for managing provenance differ greatly in various aspects, such as the degree of granularity or the models used for representing runs [CBT09; MLA+08]. Further, the concrete relationship between a workflow and its trace varies a lot due to the differences in the semantics of workflow specifications.

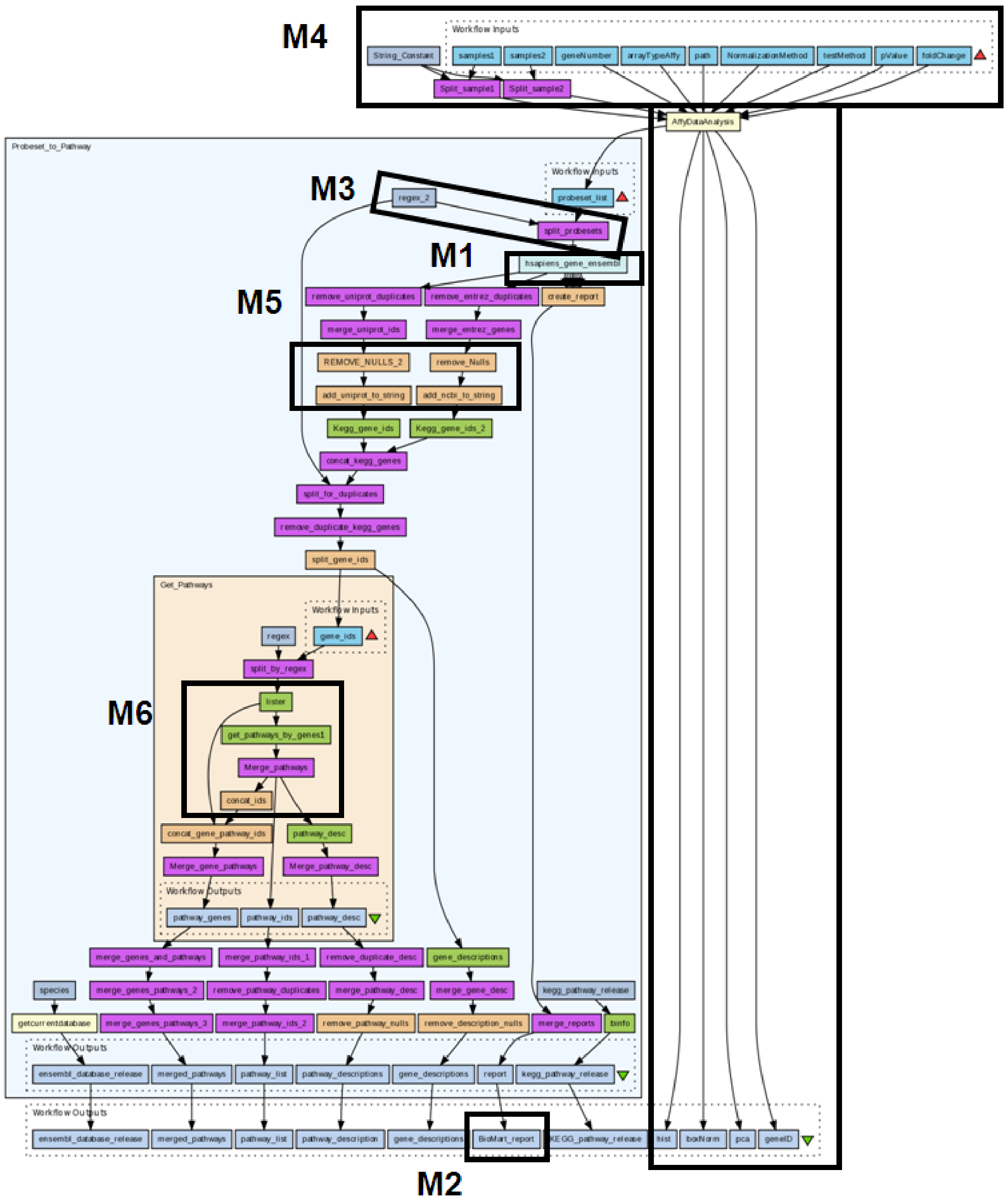

Figure 1 : One Workflow from myExperiment (W10) composed of 80 processes including 9 inputs and 11 outputs and processes of various kinds (e.g., green processes for calls to Web services, purple for local Java code) and including two sub-workflows (denoted by larger boxes in light blue and light orange) to enhance modularity and reusability.

From a user perspective, this heterogeneity is a nuisance. Although there are movements towards unifying provenance management [MLA+08], the current state-of-the-art is characterized by a high level of heterogeneity between systems, both on the trace and on the specification language. As a consequence, a true user has no way of comparing the specifications of two workflows from two given systems, nor is it possible to port a workflow from a system X to a system Y. Users are therefore bound to one specific SciWFM, which severely restricts the choice of solutions they could search in theoretically.

2.3 Workflow Repositories

Clearly, our vision of true users searching the solution to their problem among existing workflows critically depends on the existence of large repositories for workflows supporting powerful methods of search and reuse. Some such systems have already been developed, offering for the first time to the community a plethora of real examples of scientific workflows. The precursor and major project in the Life Sciences is myExperiment [DRGS09] (http://www.myexperiment.org/), a web-based public repository of workflow definitions with the explicit aim to enable sharing of those. As of June 2011, more than 1,300 workflows are freely available. Most of the workflows in the repository are from Taverna, but myExperiment also hosts some workflows from other systems. Any scientist can upload workflows in my Experiment. The site also supports the typical Web2.0-style features like registration, group-building, and biding / commenting on workflows. myExperiment already was the subject of user evaluations, though only to a limited scale and with no clear results [GDR+08].

Systems such as myExperiment are at the basis of our vision, but we strongly believe that major features and functionalities must be added to be truly useful to end-users. First, the rich structure of workflows should be considered as first-class citizens in these systems while they are currently considered only as binary black boxes with annotations. Second, searching for sub-workflows, searching based on the features of runs previously executed, sensible ranking of results, or workflow comparison (see Chapter 3 and 4 for details) should be provided. Currently, searching workflows is only possible by keywords; the repository does not offer advanced similarity search methods that could, for instance, take the topology of workflows into account (e.g. when the user wants to describe the sequence of steps to be performed).

CrowdLabs is another repository launched recently; although it is not dedicated to a particular kind of workflow, it currently only stores workflows concerned with visualization designed using the VisTrails. As for now, its community of users is smaller than the community of myExperiment. Kepler supports the installation of local repositories but offers only basic search capabilities (keyword-based search) [B. Ludäscher, personal communication]. Commercial systems like InforSense or Pipeline Pilot also offer this feature. Interestingly, we also are not aware of any comprehensive attempts in public repository building in the closely related fields of business workflows or ETL (see [Alb09; GDR+08]). This probably can be attributed to the fact that business workflows (as ETL processes) are rarely shared between communities; instead, the set of workflows of a company are a major commercial asset that is intensively curated and kept secret. Accordingly, workflow sharing has not achieved the level of importance as in an open domain such as Bioinformatics, where the exchange and publication of new methods has a long tradition [GS08].

3. Supporting True Users

The current state-of-the-art in SciWFM is largely driven by attempts to ease the development and execution of workflows. In this paper we argue that it is time to change the focus and concentrate on supporting people that do not want to develop workflows anew, but that instead are interested in getting their data analyzed with as little effort as possible and with the best methods that exist, i.e., the "true users". In this section, we describe the kind of problems those people have when they try to use a SciWFM for this purpose. The different use cases will be broken down to specific technical challenges in the next chapter. All use cases we describe are inspired by concrete analysis tasks we have encountered in various multidisciplinary Life Science projects. In particular, we will consider one main domain of application within the Life Sciences, namely, microarray data analysis. For an introduction into this field, see, for instance, [CH07; LKMS08].

We base our examples on real workflows extracted from myExperiment. Table 1 gives an overview over the selected workflows (names, main inputs and outputs, number of processes composing each workflow, sub-workflows involved, and species analyzed). Note that these workflows, as are the underlying analysis problems, are fairly complex and far from the toy examples usually discussed in the SciWFM literature. For illustration, Figure 1 shows W10 as a screenshot from the Taverna workbench.

3.1 Reusing Existing Workflows

The fact that the Life Sciences are a field in which new discoveries are achieved at an impressive speed does not imply that each discovery would have a new method accompanied. Instead, new methods are rather rare; once established, they are typically used for thousands of cases. For instance, modern gene microarrays allow measuring the expression levels of all transcripts in a given sample at once. This can, for instance, help to find genes behaving particularly under certain circumstances. Since the development of this biotechnology method in the mid 90ties, it has been applied in hundreds of labs world-wide and is an offer of many commercial biotech companies. Probably any disease by now has been studied using microarrays, and most of them multiple times [LKMS08]. The methods used in these experiments are not identical due to variations in the chips being used or the question being studied, but in general they are highly similar. For a given chip type, de-facto standards exist, though they may change over time [HSRC08].

True users of gene expression data typically focus on producing large amounts of high-quality experimental data. Having the data, they want them to be analyzed using the best possible methods. Such users would be highly interested in searching a repository of workflows for analyzing microarray data. The choice of a particular workflow would be driven by the amount of data produced, the concrete chip type that was used, and the primary biological question one has in mind. It would be the task of the repository to find all workflows matching the expressed constraints and rank them according to some criteria, such as popularity, reliability or physical location of services being used. The user would then simply choose one of the matching workflows, provide its input data, and click the "run" button. A local SciWFM should take the data and download and execute the chosen workflow, tracing all steps in a provenance model that may also be uploaded to the repository for later reuse (see below).

ID	Workflow Name	Input	Output	Species	#Proc	Sub-workflows
W10	Human Microarray CEL file to candidate pathways	Affymetrix: .cel file + normalisation method	List of the top differentially expressed genes (info on pathways and annotations on genes)	Human	80	Get_pathways, probset_to_pathways
W19	Mouse Microarray Analysis	List of Probe sets (from Affymetrix microarray)	Mapped Ids (Kegg, uniprot, ensembl)	Mouse	54	Get_pathways
W40	Microarray CEL file to candidate pathways	Affymetrix: .cel file + normalisation method	Diff. expressed genes, pathways, gene annotation, BioMart report	Human	81	Get_pathways, CandidatePathways
W79	Mapping microarrays onto pathways	sbml	image sbml model processes	*undefined*	41	writeSBML Extractgene
W142	Microarray CEL file to candidate pathways	Affymetrix: .cel file + normalisation method	Differentially expressed genes, pathways, gene annotations	Mouse	81	Get_pathways probeset_to_pw
W143	Human Microarray Analysis	List of Probe sets (from Affymetrix microarray)	Mapped IDs (Kegg, uniprot, ensembl)	Human	54	Get_pathways
W187	From cDNA Microarrays to Pathways and Abstracts	cDNA	Differentially expressed genes, pathways, pubmed report	E. coli	73	Search pubmed, retrieve abstract
W174	AffyArrayQualityAnalysis	CEL files	Quality control scores	*undefined*	25	Check_status, download_files

Table 1. Overview over the eight workflows used as example in this paper. A workflow denoted as Wx is available at http://www.myexperiment.org/workflows/x.html, where x is the id indicated on Table 1. "#Proc" gives the number of individual processes a workflow is composed of. "Sub-workflows" lists those workflows that are embedded into the main workflow by name. Names sometimes are abbreviated to save space.

Illustration: The workflows of Table 1 are all related to microarray data analysis. Some but not all of them are dedicated to Affymetrix data analysis (a given brand of chip, see column "Input"). Such workflows may still differ in (1) the format expected for the input, conforming to different levels of pre-processing of the raw data, (2) the biological question they answer and thus the kind of output they provide (pathways, gene and/or protein annotations, information about the quality of the input data), and (3) the kind of individual services they use, which can range from sequences of external services to monolithic custom-developed scripts.

Supporting a user in finding the right workflow for his purpose in this setting requires first to enable her to clearly specify their needs. This can, for instance, use keyword-based searching, specification of type-constraints on input- or output, searching with a rough idea on the types of analysis steps performed, etc. We elaborate on these options in Section 4.1

3.2 Searching and Comparing Workflows

Having performed an analysis with a concrete workflow X, many users will be interested in also trying other pipelines for the same task. This can be achieved by either repeating the steps described in the previous section, or by letting the system search for another workflow that is similar to X. Thus, given a concrete workflow X, the repository should be able to search and rank workflows that are similar to X. This requires similarity measures adapted to the domain of SciWFM: They should take into account things such as input and output types of individual tasks and of the entire workflow, the individual purpose of the tasks involved, and the order in which the tasks are to be executed, i.e., the topology of the workflow graph.

Having selected a similar workflow Y, a user often will be interested in comparing X and Y. This, again, should be supported by the repository. The search algorithm thus must be able to not only measure an abstract similarity, but also to pinpoint the concrete differences between two workflows, which is a problem of workflow alignment. Another interesting functionality is to cluster all workflows resulting from a search into groups of highly similar specifications; this would enable the user to choose truly different methods for validation runs instead of repeating again and again an almost identical workflow. Furthermore, clustering the result space is very helpful for giving an overview over the range of solutions contained in a repository.

Illustration: Let us consider that our user is working on a given disease. Using results of microarray experiments, she is interested in biological pathways that behave specifically in the diseases samples. Suppose she has used W10 for this purpose and is now looking for alternatives. Based on the kind of input/output she provides and wants to obtain, the SciWFM could suggest workflows W40, W142, W19, or W143 as possible alternatives.

A deeper look at the set of workflows S1={W10, W142,W40} shows that those workflows are rather similar in terms of structure. Figure 1 represents W10 and we have underlined (globally) the portions of W10 which are modified in the workflows described hereafter. W10 and W142 only differ in the genome used but perform the exact same series of steps, except one formatting step (M1 is replaced by one alternative module) dedicated to the species. W40 differs in the output provided (M2 is missing) and performs formatting steps on the input in a different way (M3 is replaced by several other modules branched in another part of the workflow). As a consequence, the (names of the) sub-workflows used vary: probeset_to_pathways vs candidate_pathways. Accordingly, S1 should form a separate cluster of workflows in the search result and S2 = {W19, W143} should form the other cluster. Interestingly, all together, workflows from S1 and S2 are also both structurally and intentionally very similar. They diverge all on the first processes (roughly speaking, M4 is removed is replaced by a different suite of modules) because the data format is different (list of probe sets vs /cel file). Variation between S1 and S2 also occur at the level of modules M5 and M6. Thus, in a hierarchical clustering {S1}∪{S2} may form a bigger cluster on a higher level.

While topological aspects may be important to determine workflow similarity, it is not the only feature to be considered. W187 is, for instance, close to workflows from S1 and S2 in the sense that the input/output of the main tasks (from chip data to pathways) are the same. However, it is structurally very different. This is also true for W79 that, in contrast to the other workflows, invokes complex library functions (that remain black boxes to the workflow engine) instead of separate services for separate tasks as the other workflows do. This renders, for instance, W79 to be structurally very different from W142 although the achieved functionalities are very similar.

3.3 Adapting Existing Workflows

The former two use cases aimed at finding an existing workflow for analyzing a dataset produced with an established technology. However, sometimes users have more innovative datasets whose analysis requires the adaptation of existing workflows. For instance, there exist various technological variations of gene microarrays. One example are exon arrays, where the expression of single exons instead of entire genes is measured. Measuring exon arrays is a fairly new technology, and no standards for analyzing them have emerged, yet; however, many steps from gene expression analysis can be directly reused [ZL10].

Illustration: Essentially any of the workflows described in Table 1 can be adapted to the analysis of exon arrays, though it would require different amounts of work. In any case, the first steps (normalization,,mapping probesets to transcripts) need to be adapted, while all the downstream analysis (search for affected pathways, comparison with literature etc.) can be performed in exactly the same way as with classical microarray data. Depending on the workflow, changing the first steps can be done in a few minutes or may require more work, especially when it has been implemented using custom scripts. The smallest number of modification would be necessary for W79 which is built completely on packages from Bioconductor that can work both with gene and exon level measurements.

Faced with such a novel, yet not very distinct need, a user would first perform a search as in the previous examples. However, the repository would not be able to find a complete match and thus would inform the user that some constraints of the search could not be fulfilled, implying that the workflows returned only partly match the query. These results can be a starting point for the development of a new workflow by adapting one of the results. Workflow adaptation should be supported by the system, by clearly distinguishing matched parts of the query from others, and possibly by also suggesting solutions for the unmatched parts of the query by iteratively performing individual searches on those parts. Similarly, if all matches found are too distinct from the original needs, the user may chose to pose queries for parts of his intended analysis only.

Illustration: Consider the case where a user searches for workflows taking in genomic sequences from microarray data and performing normalization and functional pathway analysis. Though all workflow of S1 are candidates for the second part of the requirements, they do not accept sequences as input. They would thus have to be adapted by adding new tasks that extract sequence information from public databases and provide a new kind of input. As a solution for this partial requirement, the repository could suggest the sub-workflow extractgene of W79; the requirements of the user could thus be covered by chaining this sub-workflow with the "downstream" parts of any of the workflows from S1.

3.4 Assembling from Partial Solutions

If the amount of necessary adaptation exceeds a certain limit, the problem slowly becomes more and more similar to that of assembling a new workflow from partial solutions. This is the most complicated scenario occurring when small subparts of the intended analysis are already available in the repository. At the same time, this use case is, in the continuum from simply reusing a workflow to developing a new workflow, the closest to the latter and thus a task that is more typical for power users than for true users. One example from transcriptomics is the emerging trend to replace hybridization of probes to an array (as used in microarrays) with directly sequencing the expressed mRNA. A comprehensive analysis pipeline for such data could take building blocks from various existing workflows (sample preparation, data normalization, sequence-to-genome mapping, sequence assembly, copy number detection etc.), but the changes to a microarray workflow are so severe that the assembly would feel much like developing a new workflow.

An intelligent repository also can be of great help in this case. For instance, the assembly can be supported by checking format and type compatibility of the different parts, by suggesting glue-code for performing necessary data transformations, or by rewriting workflows into equivalent ones by identifying and removing redundant parts stemming from merging semantically overlapping workflows.

3.5 Exploiting Information about Runs

In many of the tasks described above, users may exploit information about runs of workflows that have been executed by others or by themselves and that are stored in the repository. First, this may allow users to find workflows that have been used on very similar data (e.g., gene expression from the same tissue or species). It may also be used to point a user to workflows used on data similar to their own input. For instance, some tasks may perform better or worse when used with eukaryote or prokaryote data, two families of species that are evolutionary very distant. Additionally, accessing previous runs may help to avoid rerunning subparts of workflows that have been run with the same data and parameters already. Such repetitions are common place in bioinformatics; however, supporting them would also require storing the results of workflow tasks.

Illustration: Consider that our user is looking for a workflow for analyzing microarray data from human Liver cells. This user would be highly interested to know on which tissue types the workflows meant for human data (W10, W40 and W143) have been run in the past, and how confident the users of these pipelines were with the results.

Another common scenario in bioinformatics is the repeated running of the same workflow on the same data, only applying small changes to some parameters. Even small variations often lead to very different final outputs. For example, the set of differentially expressed genes of the same microarray will not be the same if the search is performed with different thresholds for deciding which genes are considered as "differentially expressed". In this setting, users often would like to know why the results of two runs are different in the sense that they would like to see the point in the analysis where the intermediate results started to deviate. Finding this point can be achieved by comparing both traces. Such information is crucial to choose between two conflicting results or to assess the quality of outputs.

4. Research Directions

The use cases presented in the previous chapter can be broken down to a number of technical problems which need to be solved to make such modes of usage possible, i.e., to better support true users. In this chapter, we list those technical problems, discuss existing pieces of work targeting their solution, and point to open problems and novel research directions.

4.1 Describing what Users Search

Allowing users to express what they search is a key functionality of workflow repositories. Clearly, true users will not be experts in any query language and should be able to express their request as easily as possible. It would be highly advantageous if such queries, even if expressed verbosely in first place, would internally be rewritten into a formal query language; power users may be able to query the repository directly using it. Accordingly, we believe that a continuum of specification languages should be provided depending on the level of expertise in the user and the clearness of the requirements.

At one end of the spectrum are simple keyword queries that search a textual description of a workflow. Users describe their data and intended analysis by a set of keywords that are matched against the workflows, i.e., their documentation, metadata, data types, task names etc. This is the only type of search supported in current systems [OGA+05]. An improvement to this flat style of queries is to apply keyword searches on different levels of abstraction of a workflow, as recently proposed in [LSC10]; however, finding the right level of abstraction is a non trivial task in the complex and nested workflows typically used in science. On the other end of the spectrum, repositories should support full-fledged query languages encompassing predicates for searching IO-types of tasks, the topology of a workflow, keywords in the descriptions of tasks or the entire workflow, etc. A number of proposals in this direction have recently been put forward which we discuss in Section 4.2.

Largely unexplored are solutions in between. For instance, true users often know the type and format of data they provide (like Affymetrix GeneChip raw measurements, file format: CEL) and the type of data they expect as output (list of differentially expressed genes). Accordingly, they should be able to specify such parameters as constraints for a search, while the "intermediate" bits remain unspecified or only vaguely described. Often, true users also have a coarse-grained picture of the analysis they want to perform (like: "Data should be normalized, then aggregated by sample group, then analyzed for differential expression using a statistical test with a multiple testing correction"). The topology of the workflow can be (partly) given in such descriptions. Accordingly, queries should be augmentable with partial (high level) representations of the intended analysis. We call this a workflow sketch: representing some abstract tasks to be considered, their order, or even giving hard constraints for some tasks. Works conducted on the correspondence between abstract and concrete workflow specifications (e.g., [GGW+09; LAG03]; also see Section 2.1) may provide some hints into this direction; however, their overall intention is quite different from our vision. Another interesting starting point could be visual query languages [CM90; HS08]. Furthermore, a user should be able to express preferences about the kind of tools to be considered and the data sources to be used which could use techniques as those described in [BFL+04].

4.2 Searching Workflow Specifications

4.2.1 Searching with Workflow Sketches

While models for workflow sketches and their translation into a formal query are (to our knowledge) problems that have not been tackled before, there exist several works on query languages for workflow specifications. Most existing approaches were developed for business workflows, such as the Business Process Query Language (BPQL), BPMN-Q [AS10] or BP-QL [BEKM08]. Such approaches usually retrieve workflows based on the types of inputs and outputs of tasks. BP-QL [BEKM08] also allows the definition of flow patterns in the query while BPMN-Q enables predicates on the topology of a business workflow. Whether or not these approaches also are applicable to SciWFM is an open question.

However, none of the existing approaches allow users to express arbitrary constraints on the graph structure of the workflow, for instance to explicitly search for loops or branches. Additionally, all apply strict matching techniques and cannot rank results by similarity. Furthermore, searching heterogeneous workflows (workflows defined in different languages) is a seemingly unexplored area, though heterogeneous repositories already are a reality (such as myExperiment). Essentially, languages should be able to support arbitrary combinations of syntactic and semantic predicates and should be able to either apply strict or approximate matching.

4.2.2 Searching Similar Workflows

In the previous section, we described methods to search a workflow having only a rough idea about what it should do. But, as described in Section 3.2, it is also very interesting to search workflows similar to a concrete other workflow. However, if the specification language allows forming arbitrary graphs (especially containing loops) already searching only topological identical subgraphs is equivalent to the NP-complete subgraph isomorphism problem. However, it is not yet clear whether scientific workflow graphs really are arbitrary graphs, i.e., whether the description of scientific analysis pipelines really needs all possible ways to connect tasks. Interestingly, some workflow languages lead to graph structures that have a certain form, the most important example of which are series-parallel graphs (SP). [ZCBD+09] showed that the problem of matching SP graphs can be solved in polynomial time, which opens the door to a new class of efficient matching and ranking algorithms. Finding a similar characterization for other workflow languages is an open and important problem.

Finding similar workflows is directly associated to finding a proper similarity measure for workflows. Similarity measures for workflows should consider syntactic and semantic aspects, i.e., both the graph structure and the nature of the individual tasks (including names, input and output types, etc.). Proposals for the first problem are scarce. [GCB08] described a graph matching approach to match behavioral descriptions of services, but their method does not scale to large repositories and does not provide approximate matching. Other approaches exist in the graph search community but have not yet been applied to scientific workflows [HS08]. Ideally, when comparing structures, a rich set of edit operations (e.g., add/remove/replace a flow or a task) should be considered and weighted individually. Such weights could be derived from levels of confidence in tools assigned by a user or by experts. This allows distinguishing workflows logically performing the same tasks but using tools of various levels of reliability. From the semantic point of view, the similarity measure may exploit the fact that data types (at least in the life sciences where an increasing number of tools have their input/output described through ontologies) are readily organized into hierarchies. Approaches from the data mining community (e.g., [CBGZ06] considering distances between terms within a hierarchy for matching may be used in this context.

There is also ongoing work tackling the similarity of workflows from a radically different point of view. In these works, workflows are modeled as Petri Nets and are compared by their behavior under all possible inputs [RFL06]. Whether or not such methods are applicable for scientific workflows is not yet clear.

4.2.3 Searching Sub-workflows

The former methods always compare two workflows entirely. But sometimes it is also interesting to find a workflow which contains a sub-workflow that matches best to a sub-workflow of the query (also called local alignment instead of global alignment). To our knowledge, this problem has not been addressed for workflows so far. Methods can be inspired from previous work on local alignment of strings [Gus97] and, more interestingly, on schema matching in XML [ADMR05]. Approaches like BP-QL work on BPEL specifications represented as XML documents [BEKM08] but always consider the entire workflow and only perform exact searching.

4.2.4 Clustering Workflows

In large repositories, users could benefit greatly from an intuitive structuring of potentially large result lists. Apart from ranking according to the quality of a match, results should also be grouped into clusters of similar workflows. Depending on the similarity measure, workflows can be clustered such that each workflow in a cluster essentially solves the same type of problem but uses a different approach. Such a clustering can also help in browsing by automatically structuring the space of available workflows.

The technical challenge here is probably less novel than for the other problems we describe; however, we are not aware of any existing work in this direction. Finding a proper similarity measure is, again, the central problem. Furthermore, the clustering should be able to differentiate between hard constraints (like data types to work on, expected result) and soft constraints (like concrete implementation of a particular task, topology of the data flow). Techniques for clustering database schemas [SMH+10] or structured interfaces [WYDM04] may provide hints for this problem.

4.2.5 Interactive Search

A particularly important feature would be to provide an interactive search interface. Users being presented a list of matching workflows probably have a hard time to decide which one to choose. Any support given by the system would be of great help [RSC09]. For instance, the repository could cluster search results (see Section 4.2.4), then choose one representative per cluster (e.g. the medoid workflow), and finally let the user chose between those representatives. This approach can be applied recursively, allowing for an iterative query refinement. For instance, Stoyanovich et al. recently showed how data mining techniques can be used to cluster workflows based on the keywords appearing in the workflows description [STD10]. On the same line, users could be supported by selecting discriminative properties from similar workflows (like the particular tools used for the same task) and presenting them as binary choices to the user. Such choices could be augmented with techniques from collaborative filtering (users who have used this tool have then used this other tool [SK09]). Finally, the whole range of techniques developed for incorporating user feedback into a search should be considered [GF04].

4.3 Searching Workflow Runs

This subsection provides technical foundations to the needs presented in section 3.5 in which information about the executions of workflows (the runs) need to be exploited.

4.3.1 Comparing Runs to find Points-of-Deviation

Understanding why one workflow execution resulted in a different output than the execution of a similar or even identical workflow on the same data requires finding the point-in-time where the two runs have diverged. If the runs come from completely different workflows, the problem is similar to comparing two specifications (see Section 4.2). But when the two runs obey the same workflow specification then an algorithm may exploit the known common points to guide the placement of edit operations for aligning the runs. This is, or instance, exploited in the PDiffview system [ZCBD+09]. However, it will also occur that the runs come from different yet highly similar workflows (where one was created by substituting selected tasks performing the same logical operation), in which case a hybrid approach must be followed. This problem has, to our knowledge, not yet been tackled.

4.3.2 Querying Workflow Provenance

Recently, several approaches for querying workflow runs have emerged. Projects on querying provenance ([KSB10; MPB10]) are directly tailored to a specific workflow system (Kepler and Taverna, respectively). A major drawback of these approaches is that they do not consider similar traces. In the provenance community, other interesting approaches have been suggested [KIT10], but these are only considering data provenance. Whether or not these can be extended to work on workflow provenance graphs is an open issue.

4.4 Provenance for Workflow Specifications

Scientific experiments are by nature iterative, i.e., they often consist of various trials to refine the methodology. This fact also holds for scientific workflows, which often are refined, improved, extended etc. after their first usages. Knowing the evolution of a given workflow specification allows to understand the various hypotheses that the developer has tried to check. VisTrails [SVK+08] represents a workflow and its various trial-and-error steps in a tree in which each node is one state of a given workflow, i.e., they construct a phylogeny of workflow specifications. A survey on such topics can be found in [DR09].

4.5 Workflow Environment

Having chosen a workflow suitable for his needs, any true user would like to be able to run it as easily as possible. Having a workflow server attached to the repository taking in the data is possible but we believe that such an approach would fail in practice, as scientists may not be willing to give their data out of hand. Instead, it should be possible to download and run a chosen workflow locally. Such an approach raises several issues that have not yet been addressed properly.

First, the workflow system of choice must be installed locally. Second, all services to be performed in the workflow must be installed locally or accessible from the local machine. If tasks used web services, accessibility is less of an issue, but web services are inappropriate if large quantities of data need to be analyzed. Currently, in all other cases the local and tedious installation of further software is required. But ideally, a local workflow system running a given specification should automatically download and install all tools and data sets required. To this end, SciWFM need to be equipped with mechanisms for the dynamic downloading and installation of packages. This has not yet been addressed properly in the SciWFM community, though several solutions exist in other areas (see http://www.osgi.org/).

5. Conclusions

In this paper, we argued that future development in SciWFM should concentrate on the large class of user of analysis pipelines instead of the small class of people developing such pipelines. The core of our argument is that SciWFM in fact do not offer much advantage to developers when compared to scripting languages or programming environments such as R, especially when analysis pipelines get complicated (as it tends to be the case in bioinformatics). In contrast, an intuitive representation of existing analysis workflows combined with powerful methods of searching, adapting, and running them (a repository) would unravel the full potential of SciWFM – not to be used only as a design tool, but also and most importantly as a repository of solutions and, potentially, even as an environment of their executions. We described use cases that emerge from this viewpoint and derived a number of technical challenges that need to be addressed to develop comprehensive solutions.

Please note that, most of the additional functionality we have described can be very useful to power users too. Comparing and clustering workflows will allow highlighting two very important features of workflows: (i) their decomposition in main steps and (ii) their hierarchical structure. When a new workflow has to be designed, power users may thus save time by focusing only on the new step(s) they would have to add to an existing workflow instead of having to develop a complete pipeline from scratch. Another advantage for power users are the methods to study runs; comparing workflow runs may help power users better understand why an expected output has not been produced.

We are not the first to advocate such a movement; some isolated ideas and specific aspects have been described before [DRGS09; GDR+08]. However, we are the first to give the complete picture of the potential of a *scientific workflows repository* accompanied by a break-down of use cases into technical questions and a summary on algorithms, models and tools that already exist.

There are also a number of social issues that back-up our vision.

- We believe that the existence of such repositories would increase the pressure on scientific publishers to start demanding authors to submit their programs together with their findings – in the form of workflows [SBW+09]. Enforcement of co-submitting data with papers already is common-place in some areas (especially sequences and microarrays) and in active discussion for several other types of information [CCLS08; SG07]. Note that even depositing runs is not unrealistic, as these are the ultimate proof of what was done; for instance, the large DNA sequence repositories since long allow accompanying sequences with the so-called traces which are the primary output generated from a sequencing machine. Only these traces allow fully judging of the quality of the sequences.
- As briefly stated in the introduction, a comprehensive repository also is a strong incentive for developers to properly structure and describe their solutions. This increases their visibility and, combined with typical measures of such libraries as number of visits, number of derived workflows, number of associated results, user ratings etc., also raises reputation of the authors [DRGS09]. This, in turn, requires that workflows get citable, for instance by attaching DOI's to workflows in a repository.

In essence, we advocate to share and re-use workflows. Clearly, sharing programs is not a new idea. However, it is probably safe to say this idea does not work well when targeting arbitrary software artifacts, despite decades of research in software engineering on modularization, software repositories, component-based software etc. At the same time, there are many examples where it does work perfectly, especially when the domain of the programs to share is small. Prominent examples in the Life Sciences are the Open Bio* libraries which are used in many projects (biojava, biosql, bioperl, etc.). Another example is the statistical programming environment R – today, it is a de-facto standard that new analysis methods are published as R code and that they are integrated in programming libraries such as BioConductor [GCB+04] which in turn are used around the world. Similarly, a workflow repository like those we envision must be domain specific.

6. ACKNOWLEDGMENTS

We thank Bertram Ludäscher for many helpful comments on an earlier version of this, and Johannes Starlinger and Daniela Grigori for fruitful discussions on querying workflow repositories. This work was partly funded by the Deutsche Forschungsgemeinschaft (DFG), grant GRK-SOAMED.

7. REFERENCES

[AKD10] Ailamaki, A., Kantere, V. and Dash, D. (2010). "Managing scientific data." Communications of the ACM 53(6): 68-78.

[Alb09] Albrecht, A. (2009). "METL: Managing and Integrating ETL Processes". VLDB PhD workshop.

[ADMR05] Aumueller, D., Do, H., Massmann, S. and Rahm, E. (2005). "Schema and ontology matching with COMA++". SIGMOD Conference, Baltimore, US.

[AS10] Awad, A. and Sakr, S. (2010). "Querying Graph-Based Repositories of Business Process Models ". DASFAA workshops Tsukuba, Japan.

[ZCBD+09] Bao, Z., Cohen-Boulakia, S., Davidson, S. B., Eyal, A. and Khanna, S. (2009). "Differencing Provenance in Scientific Workflows". Int. Conf. on Data Engineering. Shanghai, China.

[BEKM08] Beeri, C., Eyal, A., Kamenkovich, S. and Milo, T. (2008). "Querying business processes with BP-QL." *Information Systems* **33**(6): 477-507.

[BTN+10] Bhagat, J., Tanoh, F., Nzuobontane, E., Laurent, T., Orlowski, J., Roos, M., Wolstencroft, K., Aleksejevs, S., Stevens, R., Pettifer, S.*, et al.* (2010). "BioCatalogue: a universal catalogue of web services for the life sciences." *Nucleic Acids Res* **38 Suppl**: W689-94.

[BCB+08] Biton, O., Cohen-Boulakia, S., Davidson, S. B. and Hara, C. S. (2008). "Querying and Managing Provenance through User Views in Scientific Workflows". Int. Conf. on Data Engineering. Cancún, México.

[CCLS08] Ceol, A., Chatr-Aryamontri, A., Licata, L. and Cesareni, G. (2008). "Linking entries in protein interaction database to structured text: The FEBS Letters experiment." *FEBS Letters* **582**(8): 1171-7.

[CBGZ06] Cesa-Bianchi, N., Gentile, C. and Zaniboni, L. (2006). "Incremental Algorithms for Hierarchical Classification." *Journal of Machine Learning Research* **7**: 31-54.

[CGH+06] Churches, D., Gombas, G., Harrison, A., Maassen, J., Robinson, C., Shields, M., Taylor, I. and Wang, I. (2006). "Programming scientific and distributed workflow with Triana services." *Concurrency and Computation: Practice and Experience* **18**(10): 1021-1037.

[BFL+04] Cohen-Boulakia, S., Froidevaux, C., Lair, S., Stransky, N., Radvanyi, F., Graziani, S. and Barillot, E. (2004). "Selecting Biomedical Data Sources According To User Preferences". Int. Conference on Intelligent Systems in Molecular Biology (ISMB/ECCB), Glasgow, UK.

[CBT09] Cohen-Boulakia, S. and Tan, W.-C. (2009). Provenance in Scientific Databases. In Liu, L. and Ozsu, M. T. (ed). Book "Encyclopedia of Database Systems", Springer, pp.

[CM90] Consens, M. P. and Mendelzon, A. O. (1990). "GraphLog: a Visual Formalism for Real Life Recursion". ACM Symposium on Principles of Database Systems, Nashville, Tennessee. pp 404-416.

[CH07] Cristianini, N. and Hahn, M. W. (2007). "Introduction to Computational Genomics - A Case Study Approach", Cambridge University Press.

[DR09] Dadam, P. and Rinderle, S. (2009). Workflow Evolution In Öszu, T. and Liu, L. (ed). Book "Encyclopedia of Database Systems ", Springer, pp.**:** 3540-3544.

[DF08] Davidson, S., B. and Freire, J. (2008). "Provenance and scientific workflows: challenges and opportunities". SIGMOD Conference. Vancouver, Canada.

[DRGS09] De Roure, D., Goble, C. and Stevens, R. (2009). "The design and realisation of the Virtual Research Environment for social sharing of workflows." *Future Generation Computer Systems* **25**(5): 561-567.

[DGST09] Deelman, E., Gannon, D., Shields, M. and Taylor, I. (2009). "Workflows and e-Science: An overview of workflow system features and capabilities." *Future Generation Computer Systems* **25**(5): 528-540.

[DSS+04] Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G. B., Good, J.*, et al.* (2004). "Pegasus: A framework for mapping complex scientific workflows onto distributed systems." *Scientific Programming* **13**(3): 219-237.

[DKM+05] Delcambre, L., Kop, C., Mayr, H., Mylopoulos, J., Pastor, O., Bowers, S. and Ludäscher, B. (2005). Actor-Oriented Design of Scientific Workflows. In (ed). Book "Conceptual Modeling – ER 2005", Springer Berlin / Heidelberg, pp.**:** 369-384.

[GCB+04] Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J.*, et al.* (2004). "Bioconductor: open software development for computational biology and bioinformatics." *Genome Biol* **5**(10): R80.

[GGW+09] Gibson, A., Gamble, M., Wolstencroft, K., Oinn, T., Goble, C., Belhajjame, K. and Missier, P. (2009). "The data playground: An intuitive workflow specification environment." *Future Generation Computer Systems* **25**(4): 453-459.

[GS08] Goble, C. and Stevens, R. (2008). "State of the nation in data integration for bioinformatics." *J Biomed Inform* **41**(5): 687-93.

[GDR+08] Goderis, A., De Roure, D., Goble, C., Bhagat, J., Cruickshank, D., Fisher, P., Michaelides, D. and Tanoh, F. (2008). "Discovering scientific workflows: the myExperiment benchmarks." *IEEE Transactions on Automation Science and Engineering*.

[GNT+10] Goecks, J., Nekrutenko, A. and Taylor, J. (2010). "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences." *Genome Biol* **11**(8): R86.

[GCB08] Grigori, D., Corrales, J. C. and Bouzeghoub, M. (2008). "Behavioral matchmaking for service retrieval: application to conversation protocols." *Information Systems* **33**(7-8): 681-698.

[GF04] Grossmann, D. A. and Frieder, O. (2004). "Information Retrieval - Algorithms and Heuristics", Springer.

[Gus97] Gusfield, D. (1997). "Algorithms on Strings, Trees and Sequences", Cambridge University Press.

[HS08] He, H. and Singh, A. K. (2008). "Graphs-at-a-time: query language and access methods for graph databases". SIGMOD Conference, Vancouver, Canada pp 405-418.

[HSRC08] Ho, J. W., Stefani, M., dos Remedios, C. G. and Charleston, M. A. (2008). "Differential variability analysis of gene expression and its application to human diseases." *Bioinformatics* **24**(13): i390-8.

[HMB07] Howe, B., Maier, D. and Bright, L. (2007). "Smoothing the ROI curve for scientific data management applications". Conf. on Innovative Data Research, Asilomar, USA.

[KIT10] Karvounarakis, G., Ives, Z. G. and Tannen, V. (2010). "Querying data provenance". SIGMOD Conference, Indianapolis, US.

[KSB10] Kumar, A. M., Bowers, S. and Ludaescher, B. (2010). "Techniques for efficiently querying scientific workflow provenance graphs". Int. Conf. on Extending Database Technology. Lausanne, Switzerland.

[LSC10] Liu, Z., Shao, Q. and Chen, Y. (2010). "Searching Workflows with Hierarchical Views." *PVLDB* **3**(1): 918-927.

[LKMS08] Lottaz, C., Kostka, D., Markowetz, F. and Spang, R. (2008). "Computational diagnostics with gene expression profiles." *Methods Mol Biol* **453**: 281-96.

[LAB+05] Ludaescher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E. A., Tao, J. and Zhao, Y. (2005). "Scientific workflow management and the Kepler system." *Concurrency and Computation: Practice and Experience* **18**(10): 1039-1065.

[LAG03] Ludaescher, B., Altintas, I. and Gupta, A. (2003). "Compiling Abstract Scientific Workflows into Web Service Workflows". 15th Int. Conf. on Scientific and Statistical Database Management, Cambridge, US.

[MLB+10] Missier, P., Ludascher, B., Bowers, S., Dey, S., Sarkar, A., Shrestha, B., Altintas, I., Anand, M. K. and Goble, C. (2010).

"Linking multiple workflow provenance traces for interoperable collaborative science". 5th Workshop on Workflows in Support of Large-Scale Science, New Orleans, US.

[MPB10] Missier, P., Paton, N. W. and Belhajjame, K. (2010). "Fine-grained and efficient lineage querying of collection-based workflow provenance". 13th Int. Conf. on Extending Database Technology. Lausanne, CH.

[MLA+08] Moreau, L., Ludaescher, B., Altintas, I., Barga, R. S., Bowers, S., Callahan, S., Chin, G. J. R., Clifford, B., Cohen, S., Cohen-Boulakia, S*., et al.* (2008). "Special Issue: The First Provenance Challenge." *Concurrency and Computation: Practice and Experience* **20**(5): 409-418.

[OGA+05] Oinn, T., Greenwood, M., Addis, M., Alpdemir, M. N., Ferris, J., Glover, K., Goble, C., GODERIS, A., HULL, D., MARVIN, D*., et al.* (2005). "Taverna: Lessons in creating a workflow environment for the life sciences." *Concurrency and Computation: Practice and Experience* **18**(10): 1067-1100.

[RLS+06] Radetzki, U., Leser, U., Schulze-Rauschenbach, S. C., Zimmermann, J., Lussem, J., Bode, T. and Cremers, A. B. (2006). "Adapters, shims, and glue - service interoperability for in silico experiments." *Bioinformatics* **22**(9): 1137-43.

[RFL06] Reisig, W., Fahland, D., Lohmann, N., Massuthe, P., Stahl, C., Weinberg, D., Wolf, K. and Kaschner, K. (2006). "Analysis Techniques for Service Models". 2nd Int. Symp. on Leveraging Applications of Formal Methods, Verification and Validation. pp 11-17.

[RSC09] Ren, K., Sarvas, R. and Calic, J. (2009). "Interactive search and browsing interface for large-scale visual repositories." *Multimedia Tools and Applications* **49**(3): 513-528.

[SVK+08] Scheidegger, C., E. , Vo, H., T. , Koop, D., Freire, J. and Silva, C., T. (2008). "Querying and re-using workflows with VisTrails". SIGMOD Vancouver, Canada.

[SBW+09] Schofield, P. N., Bubela, T., Weaver, T., Portilla, L., Brown, S. D., Hancock, J. M., Einhorn, D., Tocchini-Valentini, G., Hrabe de Angelis, M. and Rosenthal, N. (2009). "Post-publication sharing of data and tools." *Nature* **461**(7261): 171-3.

[SMH+10] Seligman, L., Mork, P., Halevy, A., Smith, K., Carey, M. J., Chen, K., Wolf, C., Madhavan, J. and Kannan, A. (2010). "OpenII: An Open Source Information Integration Toolkit". Int. Conf. on Very Large Databases, Singapore.

[SG07] Seringhaus, M. R. and Gerstein, M. B. (2007). "Publishing perishing? Towards tomorrow's information architecture." *BMC Bioinformatics* **8**: 17.

[Ste08] Stein, L. D. (2008). "Towards a cyberinfrastructure for the biological sciences: progress, visions and challenges." *Nature Reviews Genetics* **9**(9): 678-88.

[STD10] Stoyanovich, J., Taskar, B. and Davidson, S. (2010). "Exploring Repositories of Scientific Workflows". Int. Workshop on Workflow Approaches to New Data-centric Science Indianapolos, US.

[SK09] Su, X. and Khoshgoftaar, T. M. (2009). "A Survey of Collaborative Filtering Techniques." *Advances in Artificial Intelligence*.

[WYDM04] Wu, W., Yu, C., Doan, A. and Meng, W. (2004). "An interactive clustering-based approach to integrating source query interfaces on the deep Web". SIGMOD Conference, Paris, France.

[ZL10] Zimmermann, K. and Leser, U. (2010). "Analysis of Affymetrix Exon Arrays". Technical Report 235, Department for Computer Science, Humboldt-Universität zu Berlin.