Scientific Workflows: Business as Usual?*

Bertram Ludäscher[1,2], Mathias Weske[3], Timothy McPhillips[1], and Shawn Bowers[1]

[1] Genome Center, University of California Davis, USA
{ludaesch,tmcphillips,bowers}@ucdavis.edu
[2] Department of Computer Science, University of California Davis, USA
[3] Hasso-Plattner-Institute University of Potsdam, Germany
weske@hpi.uni-potsdam.de

Abstract. Business workflow management and business process modeling are mature research areas, whose roots go far back to the early days of office automation systems. Scientific workflow management, on the other hand, is a much more recent phenomenon, triggered by (i) a shift towards data-intensive and computational methods in the natural sciences, and (ii) the resulting need for tools that can simplify and automate recurring computational tasks. In this paper, we provide an introduction and overview of scientific workflows, highlighting features and important concepts commonly found in scientific workflow applications. We illustrate these using simple workflow examples from a bioinformatics domain. We then discuss similarities and, more importantly, differences between scientific workflows and business workflows. While some concepts and solutions developed in one domain may be readily applicable to the other, there remain sufficiently many differences that warrant a new research effort at the intersection of scientific and business workflows. We close by proposing a number of research opportunities for cross-fertilization between the scientific workflow and business workflow communities.

1 Introduction

Whether scientists explore the limits and origins of the observable universe with ever more powerful telescopes, probe the invisibly small through particle accelerators, or investigate processes at any number of intermediate scales, scientific knowledge discovery increasingly involves large-scale data management, data analysis, and computation. With researchers now studying complex ecological systems, modeling global climate change, and even reconstructing the evolutionary history of life on Earth via genome sequencing and bioinformatics analyses, science is no longer "either physics or stamp collecting"[1]. Instead, science is increasingly driven by new and co-evolving observational and experimental methods, computer simulations, and data analysis methods. Today's scientific experiments happen in large parts *in silico*, i.e., in the computer [8]. In the UK, the term *e-Science* [1] was coined to describe computationally and data intensive science, and a large e-Science research program was started there in 2000. Similarly, in the US, the National Science Foundation created a new Office for Cyberinfrastructure (OCI) to advance computer science and informatics technologies in support

* This research was conducted while the second author was on sabbatical leave at UC Davis.

[1] *"All science is either physics or stamp collecting."* – Ernest Rutherford [11].

U. Dayal et al. (Eds.): BPM 2009, LNCS 5701, pp. 31–47, 2009.

of e-Science. As a result, the new opportunities of data-driven and compute-intensive science have introduced the new challenges of managing the enormous amounts of data generated and the complex computing environments provided by cluster computers and distributed Grid environments.

Given these developments, domain scientists face a dilemma. Scientific progress in their fields relies ever more on complex software systems, high-performance computing environments, and large-scale data management. For example, advanced computational science simulations involve all of the above [46]. But employing all of these resources is a time-consuming and labor-intensive task, made all the more challenging by the high rate at which new technologies, services, and applications appear. Understandably, many scientists would prefer to focus on their scientific research and not on issues related to the software and platforms required to perform it. As a result, interest in the area of *scientific workflow management* has increased significantly in recent years [25, 28, 30, 38–40, 49, 51], and many projects are now employing or developing scientific workflow technology [5, 19, 26, 27, 37, 45].

One goal of scientific workflows is to support and whenever possible automate what would be otherwise error-prone, repetitive tasks, e.g., data access, integration, transformation, analysis, and visualization steps [39]. Thus, scientific workflows are often used to chain together specialized applications and new data analysis methods. However, as is the case in business workflow management, scientific workflows are not only about workflow enactment and execution; modeling, design, analysis, and reuse of workflows are also becoming increasingly important in this area. The main goals of scientific workflows, then, are (i) to save "human cycles" by enabling scientists to focus on domain-specific (science) aspects of their work, rather than dealing with complex data management and software issues; and (ii) to save machine cycles by optimizing workflow execution on available resources.

In this paper, we provide an introduction and overview of scientific workflows, and compare and contrast with the well-established, mature area of business workflows. The outline and contributions of this paper are as follows. In Section 2 we provide an overview of the scientific workflow life cycle and common use cases. Section 3 describes some key concepts and emerging approaches for addressing the technical challenges encountered in developing and deploying scientific workflows. A family of bioinformatics workflows is then used in Section 4 to further illustrate some of the use cases and technical issues. In Section 5, we compare and contrast scientific workflow concepts and issues with those in found in the business workflow arena. Finally, in Section 6 we propose areas of future research and opportunities for cross-fertilization between the scientific workflow and business workflow communities.

2 The Scientific Workflow Life Cycle

Figure 1 depicts a high-level view of the scientific workflow life cycle. Starting from a scientific hypothesis to be tested, or some specific experimental goals, a **workflow design** phase is initiated. During this phase, scientists often want to reuse pre-existing workflows and templates or to refine them. Conversely, they can decide to share a (possibly revised and improved) workflow design, or make workflow products (derived data,

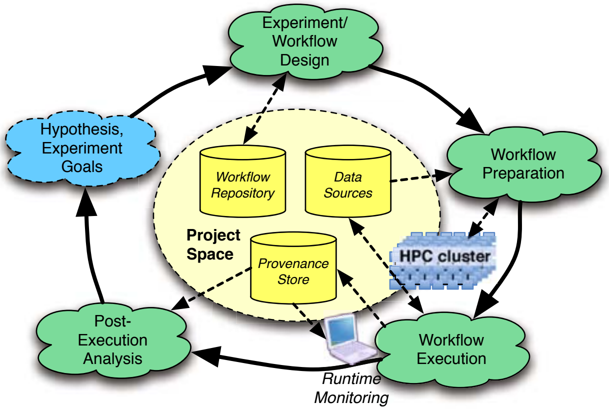


Fig. 1. Scientific Workflow Life Cycle

new components, subworkflows, *etc.*) available via a public repository or shared **project space**. Scientific workflow design differs significantly from general programming, with analysis libraries, available web services, and other pre-existing components often being "stitched together" (similar to scripting approaches [48]) to form new data analysis pipelines.

During **workflow preparation**, data sources are selected and parameters set by the user. Workflows may require the scheduling of high-performance computing (HPC) resources such as local cluster computers, or remote (Grid or cloud computing) resources; also data may have to be staged, i.e., moved to certain locations where the compute jobs running on the HPC cluster(s) expect them.

During **workflow execution**, input data is consumed and new data products created. For large-scale computational science simulations (running on hundreds or thousands of nodes; for hours, days, or weeks at a time), *runtime monitoring* is critically important: intermediate data products and special provenance information are often displayed on a web-based monitoring "dashboard" to inform the scientist about progress and possible problems during execution. Depending on this information, the scientist may decide to abort a simulation or workflow run.

Scientists often need to inspect and interpret workflow results in a **post-execution analysis** phase to evaluate data products (*does this result make sense?*), examine execution traces and data dependencies (*which results were "tainted" by this input dataset?*), debug runs (*why did this step fail?*), or simply analyze performance (*which steps took the longest time?*). Depending on the workflow outcomes and analysis results, the original hypotheses or experimental goals may be revised or refined, giving rise to further workflow (re-)designs, and a new iteration of the cycle can begin.

The workflow life cycle typically involves users in different **roles**: Domain scientists often act as the (high-level) *workflow designers* and as the *workflow operators*, i.e., they execute and possibly monitor the workflow after having prepared the run by selecting datasets and parameters. Depending on the complexity of the target workflows and the skills required to compose these in a particular system, *workflow engineers*[2] commonly are also involved in implementing the workflow design.

Types of Scientific Workflows. There is no established scientific workflow classification yet. Indeed, there seems to be no single set of characteristic features that would uniquely define what a scientific workflow is and isn't. To get a better grasp of the meaning and breadth of the term 'scientific workflow', we have identified a number of dimensions along which scientific workflows can be organized.

In many disciplines, scientists are designers and developers of new experimental protocols and data analysis methods. For example in bioinformatics, the advent of the next generation of ChIP-Seq[3] protocols and the resulting new raw data products are leading to a surge in method development to gain new knowledge from the data these experiments can produce. Scientific workflows in such realms are often *exploratory* in nature, with new analysis methods being rapidly evolved from some some initial ideas and preliminary workflow designs. In this context, it is crucial that scientific workflows be easy to reuse and modify, e.g., to replace or rearrange analysis steps without "breaking" the analysis pipeline. Once established, *production workflows*, on the other hand, undergo far fewer changes. Instead, they are executed frequently with newly acquired datasets or varying parameter settings, and are expected to run reliably and efficiently.

Scientific workflow designs can also differ dramatically in the types of steps being modeled. For example, we may distinguish *science-oriented workflows* [42], in which the named steps of the workflow spell out the core ideas of an experimental protocol or data analysis method, from lower-level *engineering* (or "plumbing") *workflows*, which deal with data movement and job management [46]. Another category along this dimension are *job-oriented workflows*, typically expressed as individual compute jobs for a cluster computer, whose job (i.e., task) dependencies are modeled as a DAG [22].

3 Scientific Workflow Concepts and System Features

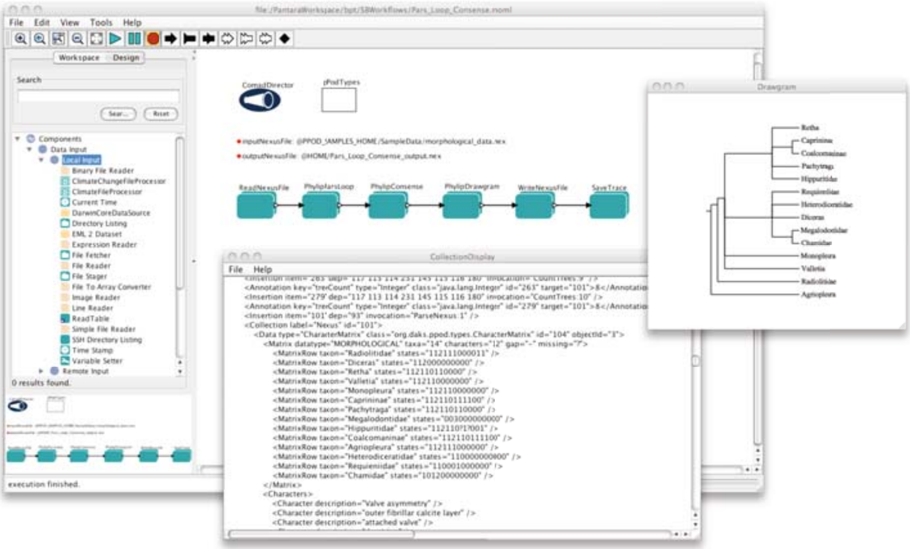
In order to address the various challenges encountered throughout the scientific workflow life cycle, and in light of the vastly different types and resulting requirements of scientific workflows, a number of concepts have been and are being developed. In the following, we use terminology and examples from the Kepler scientific workflow system [2, 37] (similar concepts exist for other systems, e.g., Taverna [3], Triana [4], *etc.*)

Integrated Workflow Environment. Many (but not all) scientific workflow systems aim at providing an integrated 'problem-solving environment'[4] to support the workflow life cycle illustrated in Figure 1. For workflow design, a *visual programming interface* is often used for wiring up reusable workflow components (or *actors*). To facilitate rapid workflow development and reuse, *actor libraries* (containing executable code) and *workflow repositories* (e.g., myExperiment [31]) can be used. Similarly, a *metadata catalog* may be used to locate relevant datasets from distributed data networks. The

[2] i.e., software engineers with workflow system expertise.

[3] **Ch**romatin **I**mmuno**P**recipitation with massively parallel DNA **Seq**uencing.

[4] A term that has been used earlier in the context of computational science simulations [47].



● inputNexusFile: @PPOD_SAMPLES_HOME/SampleData/morphological_data.nex
● outputNexusFile: @HOME/Pars_Loop_Consense_output.nex

```
<Insertion item="263" dep="117 113 114 231 145 115 116 180" invocation="CountTrees:9" />
<Annotation key="trerCount" type="Integer" class="java.lang.Integer" id="263" target="101">8</Annotatio
<Insertion item="279" dep="117 113 114 231 145 115 116 180" invocation="CountTrees:10" />
<Annotation key="trerCount" type="Integer" class="java.lang.Integer" id="279" target="101">8</Annotatio
<Insertion item="101" dep="93" invocation="ParseNexus:1" />
<Collection label="Nexus" id="101">
  <Data type="CharaterMatrix" class="org.daks.ppod.types.CharacterMatrix" id="104" objectId="3">
    <Matrix datatype="MORPHOLOGICAL" taxa="14" characters="12" gap="-" missing="?">
      <MatrixRow taxon="Radiolitidae" states="112111000011" />
      <MatrixRow taxon="Diceras" states="112000000000" />
      <MatrixRow taxon="Retha" states="112110110000" />
      <MatrixRow taxon="Valletia" states="112110000000" />
      <MatrixRow taxon="Monopleura" states="112110000000" />
      <MatrixRow taxon="Caprininae" states="112110111100" />
      <MatrixRow taxon="Pachytraga" states="112110110000" />
      <MatrixRow taxon="Megalodontidae" states="003000000000" />
      <MatrixRow taxon="Hippuritidae" states="112110?1?001" />
      <MatrixRow taxon="Coalcomaninae" states="112110111100" />
      <MatrixRow taxon="Agriopleura" states="112111000000" />
      <MatrixRow taxon="Heterodiceratidae" states="110000000800" />
      <MatrixRow taxon="Requieniidae" states="110001000000" />
      <MatrixRow taxon="Chamidae" states="101200000000" />
    </Matrix>
    <Characters>
      <Character description="Valve asymmetry" />
      <Character description="outer fibrillar calcite layer" />
      <Character description="attached valve" />
```

Fig. 2. A Kepler scientific workflow for inferring evolutionary relationships using morphological data. The windows labeled *Drawgram* and *CollectionDisplay* show the resulting phylogenetic tree and reveal the nested data streaming through the actors, respectively.

screenshot in Figure 2 depicts the Kepler user interface, including the workflow canvas and actor library.

Workflow Preparation and Execution Support. Many scientific experiments require multiple workflow runs using different parameter settings, data bindings, or analysis methods. In such cases, *parameter sweeps* [5] can be used to simplify these experiments. When running workflows repeatedly with varying parameter settings, data bindings, or alternative analysis methods, a "smart rerun" capability is often desirable [37] to avoid costly recomputation. This can be achieved, e.g., by using a data cache and analysis of dataflow dependencies (when parameters change, only downstream computations need to be re-executed [6]). For long-running workflows, some systems offer capabilities for runtime monitoring, e.g., using web-based *dashboards*, which display and visualize key variables (e.g., of large-scale fusion simulation experiments [34]). Similarly, long-running workflows require support for *fault-tolerance*, e.g., in the case of actor-, service-, or other workflow-failures, a "smart resume" capability avoids re-execution of previously successful steps, either by a form of (application-dependent) checkpointing [38, 46], or by employing suitable logging and provenance information recorded by the workflow system [21]. In so-called "grid workflows", workflow jobs need to be scheduled and mapped onto the distributed computing resources [26].

Data-Driven Models of Computation. While scientific workflow designs visually emphasize processing steps, the actual computation is often *data-driven*. Indeed, without workflow system support, scientists often spend much of their time reading,

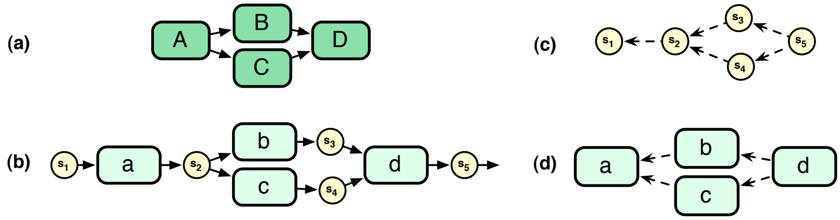


Fig. 3. Basic workflow and provenance models: (a) workflow definition (here, a DAG) with four actors A, ..., D; (b) example flow graph with process invocations a, ..., d and atomic data $s_1, \ldots, s_5$; (c) data dependencies and (d) invocation dependencies inferred from (b).

reformatting, routing, and saving datasets. Instead, *dataflow-oriented* and *actor-oriented* models of computation [36] for scientific workflows [12] emphasize the central role of data. What passes between workflow steps is not just control (the triggering of a subsequent step in response to all prior steps having completed), but *data streams*[5] that flow between and through actors (either physically or virtually [52]) and that drive the computation.

Consider the simple workflow DAG in Figure 3(a). In a business workflow model (and in some job-oriented scientific workflow models [22]), we would view A as an AND-split, followed by two task-parallel steps B and C and an AND-join D. Dataflow is often implicit or specified separately. In contrast, in a data-driven model of computation, the tokens emitted by a workflow step drive the (often repeated) invocations of downstream steps. Figure 4 illustrates that there are (often implicit) *data queues* between workflow steps to trigger multiple process invocations of an actor. Such dataflow-oriented models of computation are also beneficial for (i) *streaming workflows* (e.g., for continuous queries and window-based aggregates over sensor data streams [9]), and (ii) *pipeline-parallel* execution of scientific workflows [46]; see Figure 4 and below.

Data Provenance. In recent years, research and development activities relating to data *provenance* (or *data lineage*) and other forms of provenance information have increased significantly, in particular within the scientific workflow community [13, 18, 23, 24, 43]. Information about the processing history of a data product, especially the dependencies on other, intermediate products, workflow inputs, or parameter settings, can be valuable for the scientist during virtually all phases of the workflow life cycle, including workflow execution (e.g., for fault-tolerant [21] or optimized execution) and post-execution analysis (i.e., to validate, interpret, or debug results as described above).

Consider the *flow graph* in Figure 3(b). It captures relevant provenance information, e.g., that in a particular workflow run, the actor A consumed an input data (structure s_1) and produced output data (s_2); the linkage between inputs and outputs is given via an *invocation* a of A. The *data lineage graph* in Figure 3(c) is a view of the graph in (b), and shows how the final workflow output s_5 depends on the input s_1 via intermediate data products s_2, s_3, s_4. The *invocation dependency graph* in Figure 3(d) highlights how actor invocations depended on each other during a run.

[5] The so-called "information packets" in Flow-based Programming [44].

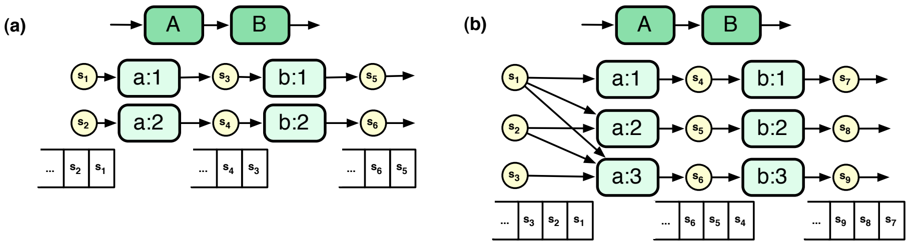


Fig. 4. The process network (PN) model supports *streaming* and *pipelined execution*: (a) step A of the workflow (top) yields two *independent* invocations (a:1, a:2) within the flow graph (bottom), possibly executed concurrently over the input stream $s_1, s_2, \ldots$; (b) a variant of (a) where A is *stateful*, preserving information between invocations a:i, resulting in additional dependencies.

Different Models of Computation and Provenance. The model of provenance (MoP) to be used for a scientific workflow may depend on the chosen model of computation (MoC) that is used to describe the workflow execution semantics [15, 41]. For example, for a simple MoC that views a workflow specification as a DAG [22], the associated MoP need not distinguish between multiple invocations a:1, a:2, ... of an actor A, simply because each actor is invoked no more than once. For the same reason, it is not meaningful to distinguish *stateless* from *stateful* actors. In contrast, in MoCs that (i) allow loops in the workflow definition, and/or (b) support pipeline parallel execution over data streams, multiple invocations need to be taken into account, and one can distinguish stateful from stateless actors. Consider Figure 4(a), which depicts a simple workflow pipeline consisting of two steps A and B. In a dataflow MoC with *firing semantics* [35], each data token s_i on a channel[6] may trigger a separate invocation; here: a:1, a:2, ..., and b:1, b:2, ... With the appropriate MoP, the provenance graph indicates that two (or more) *independent instances* of the workflow were executing. This is because the actors A and B are stateless, i.e., each invocation is independent of a prior invocation (e.g., A might convert Fahrenheit data tokens to Celsius). On the other hand, Figure 4(b) shows a provenance graph that reveals that the multiple invocations of A are *dependent* on each other, i.e., A is stateful. Such a stateful actor might, e.g., compute a running average, where a newly output token depends on more than one previously read token.

Workflow Modeling and Design. Experiment and workflow designs often start out as napkin drawings or as variants or refinements of existing workflows. Since workflows have to be executable to yield actual results, various *abstraction* mechanisms are used to deal with the complex design tasks. For example, Kepler inherits from Ptolemy II [17] the capability to nest *subworkflows* as composite actors inside of workflows, possibly adopting a different model of computation (implemented via a separate *director*) for the nested subworkflow. Top-level workflows are often coarse-grained process pipelines, where each step may be running as an independent process (e.g., executing a web service or R script), while lower-level "workflows" might deal with simple, fine-grained

[6] In the process network model [33] actors (processes) communicate via unbounded queues.

steps such as the evaluation of arithmetic expressions. Thus, it can be beneficial to employ different MoCs at different levels [46], e.g., Kahn process networks at the top-level (implemented via a so-called PN director) and synchronous dataflow[7] for lower levels.

Actor-oriented modeling and design of scientific workflows can also benefit from the use of *semantic types* [10, 12], where data objects, actors, and workflows can be annotated with terms from a controlled vocabularly or ontology to facilitate the design process. Finally, *collection-oriented modeling and design* [14] can be seen as an extension of actor-oriented modeling which takes into account the *nested collection structure* frequently found in scientific data organization to obtain workflow designs that are easier to understand, develop, and maintain [42].

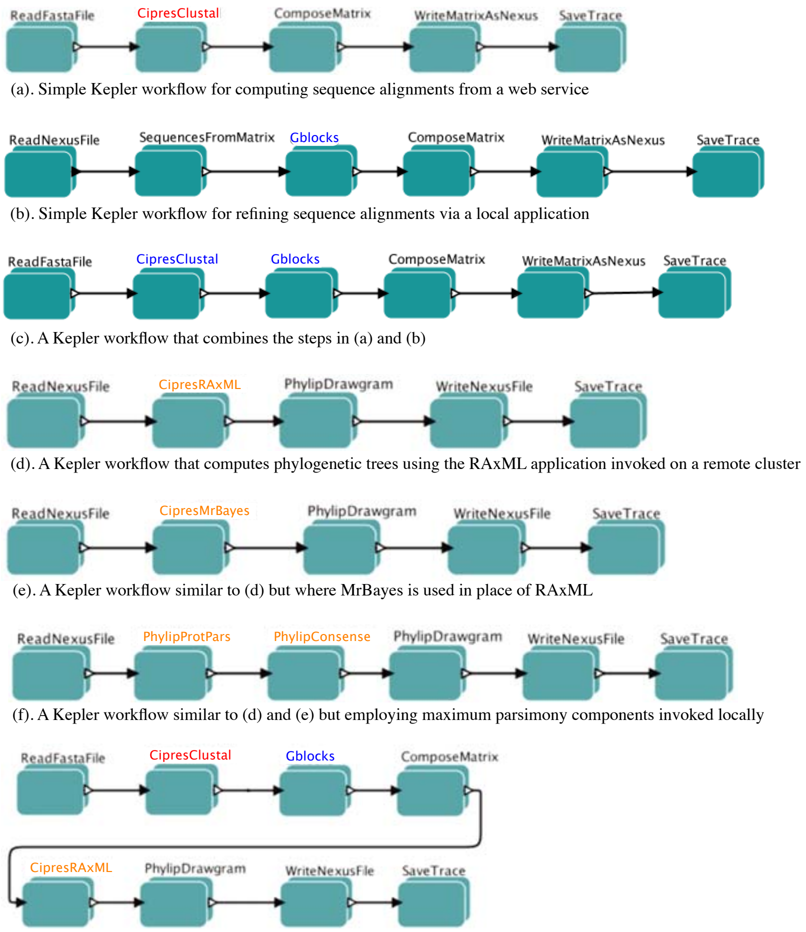
4 Case Study: Phylogenetics Workflows in Kepler

Here we illustrate the challenges, use cases, concepts, and approaches described above using concrete examples of automated scientific workflows implemented in the Kepler scientific workflow system. The example computational protocols come from the field of phylogenetics, which is the study of the tree-like, evolutionary relationships between natural groups of organisms. While phylogenetics methods and data comprise a narrow sub-domain of bioinformatics, they are broadly relevant to the understanding of biological systems in general.

The pPOD Extension to Kepler. The pPOD project[8] is addressing tool and data integration challenges within phylogenetics through a workflow automation platform that includes built-in mechanisms to record and maintain a continuous processing history for all data and computed results across multiple analysis steps. Like many other science domains, these steps currently are carried out using a wide variety of scripts, standalone applications, and remote services. Our solution, based on the Kepler system, automates common phylogenetic studies, routing data between invocations of local applications and remote services, and tracking the dependencies between input, intermediate, and final data objects associated with workflow runs [16]. The immediate goal of the current version of the system is to provide researchers an easy-to-use desktop application that enables them to create, run, and share phylogenetic workflows as well as manage and explore the provenance of workflow results. The main features of the system include: (i) a library of reusable workflow components (i.e., actors) for aligning biological sequences and inferring phylogenetic trees; (ii) a graphical workflow editor (via Kepler) for viewing, configuring, editing, and executing scientific workflows; (iii) a data model for representing phylogenetic artifacts (e.g., DNA and protein sequences, character matrices, and phylogenetic trees) that can facilitate the conversion among different data and file formats; (iv) an integrated provenance recording system for tracking data and process dependencies created during workflow execution; and (v) an interactive provenance browser for viewing and navigating workflow provenance traces (including data and process dependencies).

[7] Such subworkflows execute in a single thread, statically scheduled by an SDF director.

[8] `http://www.phylodata.org`



(a). Simple Kepler workflow for computing sequence alignments from a web service

(b). Simple Kepler workflow for refining sequence alignments via a local application

(c). A Kepler workflow that combines the steps in (a) and (b)

(d). A Kepler workflow that computes phylogenetic trees using the RAxML application invoked on a remote cluster

(e). A Kepler workflow similar to (d) but where MrBayes is used in place of RAxML

(f). A Kepler workflow similar to (d) and (e) but employing maximum parsimony components invoked locally

(g). A Kepler workflow combining the workflows in (c) and (d)

Fig. 5. Common data analyses used in phylogenetics implemented in the *Collection-Oriented Modeling and Design* (COMAD) framework of Kepler, highlighting benefits of actor reuse and workflow composition enabled through scientific workflow systems.

Figure 5 illustrates a number of workflows that can be constructed readily using the actors included with the pPOD extension to Kepler. The workflow shown in Figure 5(a) reads one or more files containing DNA or protein sequences in the FASTA file format. Each sequence represents a different group of organisms (e.g., species). The workflow then employs the Clustal application to align these sequences to each other (thereby inferring which positions in each sequence are related by evolution to positions in the other sequences). From this multiple sequence alignment, the workflow composes a phylogenetic character matrix and saves it to the researcher's disk in the Nexus format. The final actor saves a record of the workflow run containing the provenance information required to later reconstruct the derivation history of data products.

The Importance of Data Management in Tool Integration. The workflow in Figure 5(a) defines a relatively simple computation protocol: only the CipresClustal step performs a "scientifically meaningful" task, whereas the rest of the workflow simply automates the reading, reformatting, routing, and saving of data sets, and records the provenance of new data. However, even in this case, scientific workflow systems provide a number of critical benefits. In the absence of a framework such as this, researchers must run the Clustal program by hand, supplying it input data and instructions in an appropriate (but highly idiosyncratic) manner. They must either install the program on their own computers, have someone else install it for them, or else run Clustal via one of several web-based deployments of the application. In any event, they should (ideally) record precisely how they use the application each time they use it: what parameters they used, what input data files served as input, and what the immediate outputs of the program were. Each of these steps is labor-intensive and error-prone. Moreover, researchers typically carry out operations such as these *many times* on the same input data sets, varying the values of parameters given to the applications, alternately including or excluding subsets of input data sets, and repeatedly comparing and evaluating the results of all these variations. Further, because a protocol such as this occurs not in isolation, but as part of a larger set of workflows that comprise a scientific study, these kinds of variations in the upstream computational protocols cascade to the later protocols in a study, further multiplying the number of times a particular protocol must be carried out on what is conceptually the same data set. Consequently, managing data files, converting formats, and otherwise massaging scientific data in preparation for use with particular tools takes considerable time and effort, and must generally be done—again and again—by hand.

Immediate Advantages over Standard Scripting Approaches. Scientists typically define programs for automating analyses using scripting languages, e.g., when manual operation of tools such as these becomes too onerous. However, employing scripting languages in this way has serious limitations that scientific workflow systems directly aim to address. For example, a significant weakness of scripting languages is their lack of built-in provenance recording facilities. Further, the use of scripting languages for automating scientific protocols often involves *ad hoc* approaches for wrapping and executing external applications, whereas scientific workflow systems can provide users with uniform access to computational components (e.g., in Kepler through the actor model). The result is that external applications are typically only incorporated, or wrapped, into a workflow system once, making analyses easier to construct and components easier to reuse and adopt in new protocols. The limitation that scientists run into the most, however, is the difficulty of using a single script to automate a process spanning multiple compute nodes, heterogeneous communication protocols, and disparate job scheduling systems. A scientist wanting to run a scripted protocol on a local cluster rather than on her laptop must be ready to rewrite the script to take into account the associated job scheduling software, and be prepared to manually move data to and from the cluster by hand. To employ a web-based application to carry out one or more steps, she may also need to develop additional, often custom programs to send data to the service, supply it with the desired parameter values, invoke it, and wait for it to complete (which, e.g., often involves either polling or waiting for an e-mail message). Scripting languages are cumbersome platforms for this kind of distributed, heterogeneous process automation; and the situation becomes much harder when researchers wish to mix and match different kinds of applications and service protocols in a single script.

For instance, when the CipresClustal actor in Figure 5(a) is invoked, it implicitly calls a web service that runs the application remotely. By using systems such as Kepler to invoke these services, a researcher can easily repeat a protocol on the same or different data set, using all of the same parameter values used previously, or make variants of the workflows with different parameterizations. Furthermore, the researcher not only may create workflows employing multiple such services in the same workflow, but combine local applications and remote services in a single workflow. In this case, e.g., Kepler automatically routes data to and from the underlying compute resources as needed, waiting for services to complete, retrying failed service invocations, and dealing transparently with the different ways applications must be invoked on a local machine, on a Linux cluster, or at a supercomputer center.

As a further example, the workflow in Figure 5(b) can be used to refine a sequence alignment produced by the workflow in Figure 5(a) using the (locally installed) Gblocks application included with the pPod extension to Kepler. Both workflows can easily be concatenated to yield a protocol that uses heterogeneous computing resources without any effort on the part of the researcher (Figure 5c). Additional variants of workflows can easily be created without regard to how and where the particular steps of a protocol are carried out. For instance, the workflow in Figure 5(d) invokes the Cipres RAxML service at the San Diego Supercomputer Center [20] to infer phylogenetic trees from a provided character matrix, and a researcher can easily swap out this maximum likelihood method for tree inference with one based on Bayesian methods simply by replacing the CipresRAxML actor with the CipresMrBayes actor (as shown in Figure 5e). As shown, no other actors need be reconfigured. Similarly, the researcher may modify the workflow to employ a maximum parsimony method for tree inference by inserting two actors into the workflow (Figure 5f). Again, the workflow of Figure 5(d) can easily be concatenated with the workflow of Figure 5(c) to yield the workflow of Figure 5(g), which invokes two remote services and runs two local applications in the course of its execution. The ease with which new workflows can be composed, reused, repurposed, and deployed on heterogeneous resources–and later redeployed on *different* resources–is one of the major benefits of scientific workflow modeling.

5 Scientific Workflows *vs*. Business Workflows

In the following, we compare features of scientific workflows and business workflows. Even within each family, there seem to be few (if any) characteristic features that would yield a universally accepted, unambiguous classification without exceptions. Rather, it seems that workflows are related via a series of overlapping features, i.e., they exhibit a form of *family resemblance* [50]. Despite the fact that there are few sharp, categorical bounderies, the comparison below should help in assessing commonalities and typical differences between scientific workflows and business workflows.

Implementation *vs*. Modeling. The primary goal of business process modeling is to develop a common understanding of the process that involves different persons and various information systems. Once a business process model is developed and agreed upon (and in many cases improved or optimized), it can serve as a blueprint for implementing the process, all or in part, in software. Business workflows are the automated parts of these business processes. Scientific workflows, on the other hand, are developed with executability in mind, i.e., workflow designs can be viewed as executable specifications. In recent years, the modeling aspect in scientific workflows is receiving some more attention, e.g., to facilitate workflow evolution and reuse [42].

Experimental *vs*. Business-Driven Goals. A typical scientific workflow can be seen as a computational experiment, whose outcomes may confirm or invalidate a scientific hypothesis, or serve some similar experimental goals. In contrast, the outcome of a business workflow is known before the workflow starts. The goal of business workflows is to efficiently execute the workflow in a heterogeneous technical and organizational environment and, thereby, to contribute to the business goals of the company.

Multiple Workflow Instances. It is common that business workflows handle large numbers of cases and independent workflow instances at any given time. For example, each instance of an order workflow makes sure that the particular customer receives the ordered goods, and that billing is taken care of. In scientific workflows, truly independent instances are not as common. Instead, large numbers of *related* and interdependent instances may be invoked, e.g., in the context of parameter studies.

Users and Roles. Business workflows (in particular human interaction workflows) usually involve numerous people in different roles. A business workflow system is responsible for distributing *work* to the human actors in the workflow. In contrast, scientific workflows are largely automated, with intermediate steps rarely requiring human intervention. Moreover, the nature of these interactions is usually different, i.e., no work is assigned, but runtime decisions occasionally require user input (e.g., to provide an authentication information for a remote resource, an unknown parameter value, or to select from multiple execution alternatives).

Dataflow *vs*. Control-Flow Focus. An edge A → B in a business workflow typically means B can only start after A has finished, i.e., the edge represents *control-flow*. Dataflow is often implicit or modeled separately in business workflows. In contrast, A → B in a scientific workflow typically represents *dataflow*, i.e., actor A produces data that B consumes. In dataflow-oriented models of computation, execution control flows implicitly with the data, i.e., the computation is data-driven. The advantage of "marrying" control-flow with dataflow is that the resulting model is often simpler and allows stream-based, pipeline-parallel execution. The disadvantage is that certain workflow patterns (e.g., for conditional execution or exception handling) can be awkward to model via dataflow.

Dataflow Computations *vs*. Service Invocations. In scientific workflows data is often streamed through independent processes. These processes run continuously, getting input and producing output while they run. The input-output relationships of the activities are the dataflow. As a result, a sequence of actors A → B → C can provide pipelined concurrency, since they work on different data items at the same time. In business workflows, there are usually no data streams. An activity gets its input, performs some action, and produces output. An order arrives, it is checked, and given to the next activity in the process. In typical enterprise scenarios, each activity invokes a service that in turn uses functionality provided by some underlying enterprise information system.

Different Models of Computation. Different scientific workflow systems support different models of computation. For example, Pegasus/DAGMan [22, 26] workflows are job-oriented "grid workflows" and employ a DAG-based execution model without loops, in which each workflow step is executed only once. Branching and merging in these workflow DAGs corresponds to AND-splits and AND-joins in business workflows, respectively. Other workflow systems such as Taverna [3] and Triana [4] have different computation models that are dataflow-oriented and support loops; Kepler [2] supports multiple models of computation, including PN (Kahn's dataflow process network), SDF (Synchronous Dataflow, for fine-grained, single-threaded computations) and COMAD (for collection-oriented modeling and design). Given the vast range of scientific workflow types (job-oriented grid workflows, streaming workflows, collection-oriented workflows, *etc.*) there is no single best or universal model of computation that fits all needs equally. Even so, dataflow-based models are widespread among scientific workflows. In business workflows, on the other hand, Petri nets are used as the underlying foundation; BPMN is the de facto standard of an expressive process modeling language; WS-BPEL is used to specify workflows whose steps are realized by web services.

6 The Road Ahead

In this paper we have given an introduction and overview to scientific workflows, presented a bioinformatics case study, and compared features in scientific workflows with those in business workflows. Compared to the well-established area of business workflows, scientific workflow management is a fairly recent and active area of research and development.

For example, *workflow modeling and design* has not yet received the attention it deserves in scientific workflows. Workflow designs should be easy to reuse and evolve. They should be resilient to change, i.e., not break if some components are removed, added, or modified [42]. Techniques and research results from the business workflow community but also from the databases, programming languages, and software engineering communities will likely provide opportunities for future research in this area.

The business workflow community has embraced Petri nets as the unifying foundation for describing and analyzing workflows. The situation in scientific workflows is less uniform. In addition to Petri nets (e.g., combined with a complex object model [32]), there are other underlying models, e.g., well-established formalisms such as dataflow process networks [33, 35, 36], and new, specialized dataflow extensions, e.g., for nested data [42]. For optimizing streaming workflows, techniques from the database community for efficiently querying data streams look promising as well.

A very active area of research in scientific workflows is *provenance*, in particular techniques for capturing, storing, and querying not only data provenance [7] but also workflow evolution provenance (a form of versioning for configured workflows) [29]. In this context, statically analyzable dependencies between steps in a workflow can be used, for instance, to optimize data routing [52], or to check whether each step will eventually receive the required data. This is interesting, since the business workflow community has developed a set of soundness criteria for a given process model based on control-flow, disregarding data dependencies to a large extent. The integration of workflow analysis methods based on dataflow and on control-flow is a promising new area of research and cross-fertilization between the communities that can yield new results and insights for both scientific workflows and business workflows.

Acknowledgements. Work supported through NSF grants IIS-0630033, OCI-0722079, IIS-0612326, DBI-0533368, ATM-0619139, and DOE grant DE-FC02-01ER25486.

References

1. Defining e-Science (2008), `www.nesc.ac.uk/nesc/define.html`
2. The Kepler Project (2008), `www.kepler-project.org`
3. The Taverna Project (2008), `www.mygrid.org.uk/tools/taverna`
4. The Triana Project (2008), `www.trianacode.org`
5. Abramson, D., Enticott, C., Altinas, I.: Nimrod/K: Towards Massively Parallel Dynamic Grid Workflows. In: ACM/IEEE Conference on Supercomputing (SC 2008). IEEE Press, Los Alamitos (2008)
6. Altintas, I., Barney, O., Jaeger-Frank, E.: Provenance collection support in the Kepler scientific workflow system. In: Moreau, L., Foster, I. (eds.) IPAW 2006. LNCS, vol. 4145, pp. 118–132. Springer, Heidelberg (2006)
7. Anand, M., Bowers, S., McPhillips, T., Ludäscher, B.: Exploring Scientific Workflow Provenance Using Hybrid Queries over Nested Data and Lineage Graphs. In: Intl. Conf. on Scientific and Statistical Database Management (SSDBM), pp. 237–254 (2009)
8. Anderson, C.: The End of Theory: The Data Deluge Makes the Scientific Method Obsolete. WIRED Magazine (June 2008)
9. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: PODS, pp. 1–16 (2002)
10. Berkley, C., Bowers, S., Jones, M., Ludäscher, B., Schildhauer, M., Tao, J.: Incorporating Semantics in Scientific Workflow Authoring. In: 17th Intl. Conference on Scientific and Statistical Database Management (SSDBM), Santa Barbara, California (June 2005)
11. Birks, J.B.: Rutherford at Manchester. Heywood (1962)
12. Bowers, S., Ludäscher, B.: Actor-oriented design of scientific workflows. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 369–384. Springer, Heidelberg (2005)
13. Bowers, S., McPhillips, T., Ludäscher, B., Cohen, S., Davidson, S.B.: A model for user-oriented data provenance in pipelined scientific workflows. In: Moreau, L., Foster, I. (eds.) IPAW 2006. LNCS, vol. 4145, pp. 133–147. Springer, Heidelberg (2006)

14. Bowers, S., McPhillips, T., Wu, M., Ludäscher, B.: Project histories: Managing data provenance across collection-oriented scientific workflow runs. In: Cohen-Boulakia, S., Tannen, V. (eds.) DILS 2007. LNCS (LNBI), vol. 4544, pp. 122–138. Springer, Heidelberg (2007)
15. Bowers, S., McPhillips, T.M., Ludäscher, B.: Provenance in Collection-Oriented Scientific Workflows. In: Moreau, Ludäscher [43]
16. Bowers, S., McPhillips, T., Riddle, S., Anand, M.K., Ludäscher, B.: Kepler/pPOD: Scientific workflow and provenance support for assembling the tree of life. In: Freire, J., Koop, D., Moreau, L. (eds.) IPAW 2008. LNCS, vol. 5272, pp. 70–77. Springer, Heidelberg (2008)
17. Brooks, C., Lee, E.A., Liu, X., Neuendorffer, S., Zhao, Y., Zheng, H.: Heterogeneous Concurrent Modeling and Design in Java (Volume 3: Ptolemy II Domains). Technical Report No. UCB/EECS-2008-37 (April 2008)
18. Cheney, J., Buneman, P., Ludäscher, B.: Report on the Principles of Provenance Workshop. SIGMOD Record 37(1), 62–65 (2008)
19. Churches, D., Gombas, G., Harrison, A., Maassen, J., Robinson, C., Shields, M., Taylor, I., Wang, I.: Programming Scientific and Distributed Workflow with Triana Services. In: Fox, Gannon [28]
20. Cyberinfrastructure for Phylogenetic Research, CIPRES (2009), www.phlyo.org
21. Crawl, D., Altintas, I.: A provenance-based fault tolerance mechanism for scientific workflows. In: Freire, J., Koop, D., Moreau, L. (eds.) IPAW 2008. LNCS, vol. 5272, pp. 152–159. Springer, Heidelberg (2008)
22. Directed Acyclic Graph Manager, DAGMan (2009), www.cs.wisc.edu/condor/dagman
23. Davidson, S.B., Boulakia, S.C., Eyal, A., Ludäscher, B., McPhillips, T.M., Bowers, S., Anand, M.K., Freire, J.: Provenance in Scientific Workflow Systems. IEEE Data Eng. Bull. 30(4), 44–50 (2007)
24. Davidson, S.B., Freire, J.: Provenance and Scientific Workflows: Challenges and Opportunities (Tutorial Notes). In: SIGMOD (2008)
25. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems 25(5), 528–540 (2009)
26. Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G.B., Good, J., Laity, A., Jacob, J., Katz, D.: Pegasus: A framework for mapping complex scientific workflows onto distributed systems. Scientific Programming 13(3), 219–237 (2005)
27. Fahringer, T., Prodan, R., Duan, R., Nerieri, F., Podlipnig, S., Qin, J., Siddiqui, M., Truong, H., Villazon, A., Wieczorek, M.: ASKALON: A grid application development and computing environment. In: IEEE Grid Computing Workshop (2005)
28. Fox, G.C., Gannon, D. (eds.): Concurrency and Computation: Practice and Experience. Special Issue: Workflow in Grid Systems, vol. 18(10). John Wiley & Sons, Chichester (2006)
29. Freire, J.-L., Silva, C.T., Callahan, S.P., Santos, E., Scheidegger, C.E., Vo, H.T.: Managing rapidly-evolving scientific workflows. In: Moreau, L., Foster, I. (eds.) IPAW 2006. LNCS, vol. 4145, pp. 10–18. Springer, Heidelberg (2006)
30. Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J.: Examining the Challenges of Scientific Workflows. Computer 40(12), 24–32 (2007)
31. Goble, C., Roure, D.D.: myExperiment: Social Networking for Workflow-Using e-Scientists. In: Workshop on Workflows in Support of Large-Scale Science, WORKS (2007)

32. Hidders, J., Kwasnikowska, N., Sroka, J., Tyszkiewicz, J., den Bussche, J.V.: DFL: A dataflow language based on Petri nets and nested relational calculus. Information Systems 33(3), 261–284 (2008)
33. Kahn, G.: The Semantics of a Simple Language for Parallel Programming. In: Rosenfeld, J.L. (ed.) Proc. of the IFIP Congress 74, pp. 471–475. North-Holland, Amsterdam (1974)
34. Klasky, S., Barreto, R., Kahn, A., Parashar, M., Podhorszki, N., Parker, S., Silver, D., Vouk, M.: Collaborative Visualization Spaces for Petascale Simulations. In: Intl. Symposium on Collaborative Technologies and Systems (CTS), May 2008, pp. 203–211 (2008)
35. Lee, E.A., Matsikoudis, E.: The Semantics of Dataflow with Firing. In: Huet, G., Plotkin, G., Lévy, J.-J., Bertot, Y. (eds.) From Semantics to Computer Science: Essays in memory of Gilles Kahn. Cambridge University Press, Cambridge (2008)
36. Lee, E.A., Parks, T.M.: Dataflow Process Networks. Proceedings of the IEEE, 773–799 (1995)
37. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific Workflow Management and the Kepler System. Concurrency and Computation: Practice & Experience 18(10), 1039–1065 (2006)
38. Ludäscher, B., Altintas, I., Bowers, S., Cummings, J., Critchlow, T., Deelman, E., Freire, J., Roure, D.D., Goble, C., Jones, M., Klasky, S., Podhorszki, N., Silva, C., Taylor, I., Vouk, M.: Scientific Process Automation and Workflow Management. In: Shoshani, A., Rotem, D. (eds.) Scientific Data Management: Challenges, Existing Technology, and Deployment. Chapman and Hall/CRC (to appear, 2009)
39. Ludäscher, B., Bowers, S., McPhillips, T.: Scientific Workflows. In: Özsu, M.T., Liu, L. (eds.) Encyclopedia of Database Systems. Springer, Heidelberg (to appear, 2009)
40. Ludäscher, B., Goble, C. (eds.): ACM SIGMOD Record: Special Issue on Scientific Workflows, vol. 34(3) (September 2005)
41. Ludäscher, B., Podhorszki, N., Altintas, I., Bowers, S., McPhillips, T.M.: From computation models to models of provence: The RWS approach, vol. 20(5), pp. 507–518
42. McPhillips, T., Bowers, S., Zinn, D., Ludäscher, B.: Scientific Workflow Design for Mere Mortals. Future Generation Computer Systems 25, 541–551 (2009)
43. Moreau, L., Ludäscher, B. (eds.): Concurrency and Computation: Practice & Experience – Special Issue on the First Provenance Challenge. Wiley, Chichester (2007)
44. Morrison, J.P.: Flow-Based Programming – A New Approach to Application Development. Van Nostrand Reinhold (1994), www.jpaulmorrison.com/fbp
45. Oinn, T., Greenwood, M., Addis, M., Alpdemir, M.N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M.R., Senger, M., Stevens, R., Wipat, A., Wroe, C.: Taverna: Lessons in Creating a Workflow Environment for the Life Sciences. In: Fox, Gannon [28]
46. Podhorszki, N., Ludäscher, B., Klasky, S.A.: Workflow automation for processing plasma fusion simulation data. In: Workshop on Workflows in Support of Large-Scale Science (WORKS), pp. 35–44. ACM Press, New York (2007)
47. Rice, J.R., Boisvert, R.F.: From Scientific Software Libraries to Problem-Solving Environments. IEEE Computational Science & Engineering 3(3), 44–53 (1996)
48. Stajich, J.E., Block, D., Boulez, K., Brenner, S.E., Chervitz, S.A., Dagdigian, C., Fuellen, G., Gilbert, J.G., Korf, I., Lapp, H., Lehvaslaiho, H., Matsalla, C., Mungall, C.J., Osborne, B.I., Pocock, M.R., Schattner, P., Senger, M., Stein, L.D., Stupka, E., Wilkinson, M.D., Birney, E.: The BIOPERL Toolkit: Perl Modules for the Life Sciences. Genome Res. 12(10), 1611–1618 (2002)

49. Taylor, I., Deelman, E., Gannon, D., Shields, M. (eds.): Workflows for e-Science: Scientific Workflows for Grids. Springer, Heidelberg (2007)
50. Wittgenstein, L.: Philosophical Investigations. Blackwell Publishing, Malden (1953)
51. Yu, J., Buyya, R.: A Taxonomy of Scientific Workflow Systems for Grid Computing. In: Ludäscher, Goble [40]
52. Zinn, D., Bowers, S., McPhillips, T., Ludäscher, B.: X-CSR: Dataflow Optimization for Distributed XML Process Pipelines. In: 25th Intl. Conf. on Data Engineering (ICDE), Shanghai, China (2008)