Scientific Workflows: Moving Across Paradigms

CHEE SUN LIEW, University of Malaya
MALCOLM P. ATKINSON and MICHELLE GALEA, University of Edinburgh
TAN FONG ANG, University of Malaya
PAUL MARTIN, University of Amsterdam
JANO I. VAN HEMERT, Optos Plc

Modern scientific collaborations have opened up the opportunity to solve complex problems that require both multidisciplinary expertise and large-scale computational experiments. These experiments typically consist of a sequence of processing steps that need to be executed on selected computing platforms. Execution poses a challenge, however, due to (1) the *complexity and diversity of applications*, (2) the *diversity of analysis goals*, (3) the *heterogeneity of computing platforms*, and (4) the *volume and distribution of data*.

A common strategy to make these in silico experiments more manageable is to model them as *workflows* and to use a workflow management system to organize their execution. This article looks at the overall challenge posed by a new order of scientific experiments and the systems they need to be run on, and examines how this challenge can be addressed by workflows and workflow management systems. It proposes a taxonomy of workflow management system (WMS) characteristics, including aspects previously overlooked. This frames a review of prevalent WMSs used by the scientific community, elucidates their evolution to handle the challenges arising with the emergence of the "fourth paradigm," and identifies research needed to maintain progress in this area.

Categories and Subject Descriptors: A.1 [**General Literature**]: Introductory and Survey; C.1.4 [**Parallel Architectures**]: Distributed Architectures; D.2.11 [**Software Engineering**]: Software Architectures; H.4.1 [**Information Systems Applications**]: Office Automation—*Workflow management*

General Terms: Algorithms, Design, Language, Measurement, Performance

Additional Key Words and Phrases: Data-intensive science, workflows, workflow management systems

ACM Reference Format:
Chee Sun Liew, Malcolm P. Atkinson, Michelle Galea, Tan Fong Ang, Paul Martin, and Jano I. Van Hemert. 2016. Scientific workflows: Moving across paradigms. ACM Comput. Surv. 49, 4, Article 66 (December 2016), 39 pages.
DOI: http://dx.doi.org/10.1145/3012429

1. INTRODUCTION

Computational science has increasingly stood alongside *experimental* and *theoretical* science in scientific discovery over the last five decades. However, the phrase "*fourth*

Authors' addresses: C. S. Liew and T. F. Ang, Faculty of Computer Science & Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia; email: csliew@um.edu.my; angtf@um.edu.my; M. P. Atkinson, School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, UK; email: Malcolm.Atkinson@ed.ac.uk; M. Galea, The Data Lab, University of Edinburgh, 15 South College Street, Edinburgh, EH8 9AA; email: michelle.galea@ed.ac.uk; P. Martin, Informatics Institute, University of Amsterdam, Science Park 904 1098XH Amsterdam, Netherlands; email: p.w.martin@uva.nl; J. I. van Hemert, Optos, Queensferry House, Carnegie Campus, Enterprise Way, Dunfermline KY11 8GR, UK; email: jvanhemert@optos.com.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2016 ACM 0360-0300/2016/12-ART66 $15.00
DOI: http://dx.doi.org/10.1145/3012429

paradigm" was coined by Gray [2009] to draw attention to a new methodology in science that complements those three paradigms—one that addresses the increasing importance of digital data in science.

Advances in computing technology foster the use of simulations to perform complex analyses in theoretical modeling; these simulations generate large volumes of data, which are stored in databases and files. At the same time, the revolution in digital technology has increased the volume of observational data used in experimental science as the extensive use of digital sensors has been coupled with highly automated data collection, for example, digital imaging devices in astronomy and microarray DNA sequencers in genomics [Interagency Working Group on Digital Data 2009]. The scientific community is facing a massive data challenge, such as petabytes of live data streams[1] and petabytes of curated data.[2] The complexity and diversity of data pertinent to research topics is also increasing rapidly; for example, ELIXIR-supporting European life scientists host 24 curated reference data collections, each of which evolves and grows rapidly.[3]

With the 21st century, the fourth paradigm has emerged, known as *data-intensive science* [Hey et al. 2009] or *data-driven science*, where scientists discover new knowledge by systematically processing large volumes or complex collections of data captured in experiments or generated by simulations. As Jim Gray observed [Gray 2009], most astronomers do not look through the sophisticated and expensive new telescopes; instead, they work at the end of a data pipeline, analyzing derived information on their own workstations. Computing software is used extensively to integrate and analyze data in order to extract new knowledge. Data-driven science does not replace existing scientific methods; it complements existing paradigms—an iterative cycle to link knowledge with observations [Kell and Oliver 2004].

The examples that follow highlight some projects from various scientific domains that are dealing with large-scale distributed data:

Optical astronomy. The Pan-STARRS project[4] for detecting potentially hazardous objects in the solar system is equipped with four 1.4 Gigapixel resolution digital cameras that capture more than 1PB of raw data and generate 100TB data within its catalog database each year. Every day, a Load workflow creates about 700 new Load databases storing nightly detected objects, and once a week, a Merge workflow merges 50,000 Load databases with 12 offline Cold databases using Trident [Simmhan et al. 2009]. These data may be analyzed directly or used in combination with other observations, using standards mediated by the IVOA.[5]

Radio astronomy. LOFAR,[6] for observing the universe using very low-frequency radio telescopes, is producing high-quality interferometric data on baselines ranging from 100m up to more than 1,000km, from 24 core stations (within a 2km radius in The Netherlands), 16 remote stations (within 100km), and eight international stations (including France, Germany, Sweden, and the United Kingdom) [Heald et al. 2011]. The data processing pipeline involves correlating and reducing data from all of the stations connected through a wide-area network using an IBM Blue Gene/P supercomputer; real-time analysis and model tuning using a general-purpose cluster; temporary storage of the raw data; and archival of final

[1]The SKA (www.skatelescope.org) will generate 2.5 to 7.5PB of raw data/second [Broekema et al. 2012].
[2]The LHC (cms.web.cern.ch) preserves 30PB of data per year [Chalmers 2014] and the Large Synoptic Survey Telescope (www.lsst.org) will generate several petabytes of saved images and catalogs every year.
[3]ELIXIR services: https://www.elixir-europe.org/services.
[4]Panoramic Survey Telescope & Rapid Response System (Pan-STARRS): pan-starrs.ifa.hawaii.edu.
[5]International Virtual Observatory Alliance (IVOA): www.ivoa.net.
[6]Low Frequency Array (LOFAR): www.lofar.org.

data products for further use [Romein et al. 2011]—this is both data intensive and computationally complex.

Seismology. The VERCE project[7] is delivering an e-Science environment to the seismological community to exploit the increasingly large volume of seismological data. It provides an integrated architecture for diverse data-intensive applications in data analysis and modeling and the interconnection of community data infrastructures with HPC infrastructures [Atkinson et al. 2015]. This is a framework for executing heterogenous tasks that process large volumes of data (e.g., 100TB of raw data to analyze the 2011 Tōhoku earthquake and 5PB of simulation results to model the corresponding subsurface processes), from geographical distributed and diverse data sources, on grid, cloud, and HPC computing resources.

Experimental biology. OME[8] provides flexible data management and interoperability tools for biological light microscopy that deal with over 150 microscopy file formats and distributed image processing. Its OMERO project [Allan et al. 2012] provides tools for extracting measurements from microscopy images. OMERO uses multiple storage schemes (i.e., binary image repositories, relational databases, and HDF5[9]), middleware, and client applications (i.e., scripts written in Java, C, and Python, and for web browsers) to enable diverse and complex biomedical research.

Environmental science. The study of the pattern of bird species occurrence to understand how they are influenced by environmental changes [Kelling et al. 2013] is data intensive. It involves merging data from different organizations (e.g., NASA,[10] USGS,[11] NOAA,[12] AKN,[13] and citizen scientists), using a high-performance computing infrastructure to explore complex models and large volumes of data through statistical analyses and visualizations, using VisTrails [Callahan et al. 2006].

These projects, like many others, involve the challenges of data creation, exploration, exploitation, and preservation in many scientific communities. The rapidly growing and diverse data opens many new opportunities in business, research, design, policy formulation, and decision making, but these opportunities can only be exploited if we improve our knowledge discovery apparatus as we enter the data-intensive era.

Managing the data deluge not only requires larger storage space and more computational power but also demands new advances (e.g., scalable data processing algorithms that can handle massive datasets), new data management technologies for distributed and heterogeneous data sources, and new high-speed networks for transferring large volumes of data [Gorton et al. 2008]. Many datasets (e.g., three-dimensional spatial time-series data in seismology) may be stored in DBMSs designed for efficient transaction processing and not for scientific data. Boncz et al. [2008] discuss how they redesigned the database architecture in MonetDB, making use of modern technology to avoid the performance bottleneck in main-memory access, while Stonebraker et al. [2009] have specified a common set of requirements for new scientific database systems (e.g., a new array data model and operators to process time series). Budavári et al. [2013] have developed the SkyQuery Language, extending SQL to better express every aspect of cross-identification problems in large-scale astronomy archives.

[7]Virtual Earthquake and seismology Research Community e-science environment in Europe: www.verce.eu.
[8]Open Microscopy Environment (OME): www.openmicroscopy.org.
[9]HDF5: www.hdfgroup.org/HDF5.
[10]National Aeronautics and Space Administration (NASA): www.nasa.gov.
[11]United States Geological Survey (USGS): www.usgs.gov.
[12]National Oceanic and Atmospheric Administration (NOAA): www.noaa.gov.
[13]Avian Knowledge Network (AKN): www.avianknowledge.net.

The growing wealth of data (with increasing diversity and complexity across all domains) is not the only challenge that the scientific community is facing. Another challenge is the complexity and the heterogeneity of the computing systems that support the experiments, the applications, and the data. Some efforts are underway that attempt cross-architectural implementation (e.g., Grid/Cloud [Deelman 2010; Deelman et al. 2016; Kacsuk et al. 2014]), but their integration processes are proving challenging. There are a number of reasons for this:

—It is not unusual to run experiments that read raw data from distributed file systems, metadata from the databases, and live data streams from remote sensors. When collaborative work is involved, these resources may not be located at one site or managed by a single organisation. The data integration process needs to deal with different resource types and with a variety of access constraints.

—Even if the experiment only involves data stored in a file system, there are different storage solutions available. The Sphere parallel data processing engine can efficiently perform massive parallel in-storage data processing on data stored in the Sector file system (twice as fast as Hadoop MapReduce [Gu and Grossman 2009]). However, it cannot process data stored on a Gfarm file system.[14]

—There is a broad spectrum of applications, from arithmetically intensive to data intensive. Each type of application is suitable to run on certain hardware architectures. For instance, a commodity cluster provides high computing power with hundreds to hundreds of thousands of cores and usually is intrinsically attached to a storage area network to store the data. This architecture is adapted to solve compute-intensive problems. However, running data-intensive applications often incurs higher communication costs and achieves lower performance because disk I/O rates and network bandwidths become performance bottlenecks. In this case, data-intensive computing machines, as described in Dobos et al. [2013], Givelberg et al. [2011], and Norman and Snavely [2010], outperform commodity clusters.

—The execution context itself differs. For instance, Pegasus is a popular workflow management system (WMS) used to manage the execution of in silico experiments. It works well with DAGMan and HTCondor[15] [Litzkow et al. 1988; Thain et al. 2005] handling batch processing, which stages in data and executable script onto a HPC cluster and stages out the results after each task has been executed. In some contexts, many of the functions a data-driven researcher requires are packaged as web services, for example, access to curated data collections (see ELIXIR earlier) or standard transformations. Some WMSs, such as Taverna [Wolstencroft et al. 2013], are designed to orchestrate the use of such services. A way to exploit coarse-grained interoperability is to treat the workflows as "black boxes" and orchestrate them by nesting WMSs, as in the SHIWA platform [Korkhov et al. 2013].

This complexity and heterogeneity cannot be eliminated by unification of technologies as there are powerful drivers for continued diversity. Forcing a community to abandon their *existing investments* and converge on a common technology is unacceptable as it would destroy their research momentum. Much pooled intellectual effort and funds are spent over many years to develop the operational practices and their associated data interchange standards. When boundary-crossing research links two such "islands" of homogeneity, neither can afford to disrupt its community to align with the other. Some legacy systems are hard to replace or too expensive (e.g., methods use programs written decades ago); it is infeasible to marshal experts to rewrite them.

[14]Gfarm file system: datafarm.apgrid.org.
[15]HTCondor was known as "Condor" from 1988 until its name changed in 2012.

Even if a community were to agree on a standard technology, the diversity will eventually reappear due to the *independent evolution* of technology in separate groups. The Swift system was developed by the GriPhyN Virtual Data System (VDS)[16]—a collaboration to automate the analysis of the large quantities of high-energy physics data via a set of workflow tools. Initially, VDS used the Chimera virtual data language [Foster et al. 2002] to express the logical organization of operations, Pegasus (see Section 3.1) as its workflow planner, and HTCondor DAGMan as its execution engine. The Swift system (see Section 3.4) has since grown to be a stand-alone workflow system for petascale parallel execution [Wilde et al. 2009], using its own SwiftScript for iterative operations and Falkon [Raicu et al. 2007] for task submission.

The third factor sustaining complexity and diversity is the *socioeconomic power of identity*. Cloud computing [Armbrust et al. 2010] has emerged as a new paradigm that provides dynamic and scalable infrastructure for applications, computing, and storage. The key players in the industry have shown their interest and have populated this niche in the Internet ecosystem, for example, Amazon,[17] Google,[18] Microsoft,[19] and Rackspace.[20] Each has its own strengths and market share. Brynjolfsson et al. [2010] examine the cloud computing model in comparison with other utility models, such as electricity, and conclude that cloud offerings will not be interchangeable across providers. This is currently a barrier for cross-platform experiments. Juve and Deelman [2010] discuss how the scientific communities may adapt cloud computing technologies, which primarily target business needs. Zhao et al. [2015] identify the challenges of such adaptation and share their experience in integrating the Swift into the cloud. Cała et al. [2016] discuss their experience in porting a life science workflow onto Microsoft Azure cloud and provided a balanced view of the key benefits and drawbacks we observed during the migration. We argue in Section 4.1 that cross-platform working and interoperation between workflows encoded in different notations should be facilitated.

Section 2 reviews the established characteristics of workflows from a data-intensive viewpoint. It then discusses architectures for providing workflows and draws attention to some features not normally considered in order to establish a framework for discussing workflow systems. Using this framework, Section 3 analyzes six workflow systems and their utility for data-intensive scientific research. It concludes with a summary and an assessment of their data handling and optimization strategies. Section 4 charts the anticipated development of scientific workflow languages as they handle more computation and much more data. Three topics are addressed: (1) how to transcend technical and cultural boundaries while respecting community and individual needs; (2) how to empower scientists so that they can drive their own research agenda, only calling on other experts exceptionally; and (3) possible technical developments taking account of external influences and trends. This anticipates a more complex and integrated context for scientific workflows with strong influences from advances in the ways in which scientific data are stored and organized. The concluding section finishes with a clarion call for a combined effort, not only from the scientific workflow community but also from the scientific data storage, archiving, and curation communities, to develop an integrated approach to facilitating the fourth paradigm. This will require a formal framework, a pervasive campaign establishing interchange standards, and many carefully integrated advances in the engineering underpinning data handling, data organization, and their interplay with workflow enactment. Above all, it will

[16]GriPhyN VDS: www.ci.uchicago.edu/wiki/bin/view/VDS/VDSWeb/WebMain.
[17]Amazon Elastic Compute Cloud (Amazon EC2): aws.amazon.com/ec2/.
[18]Google App Engine: www.google.com/apps.
[19]Microsoft Windows Azure: www.microsoft.com/windowsazure/.
[20]Rackspace Cloud: www.rackspace.com/cloud/.

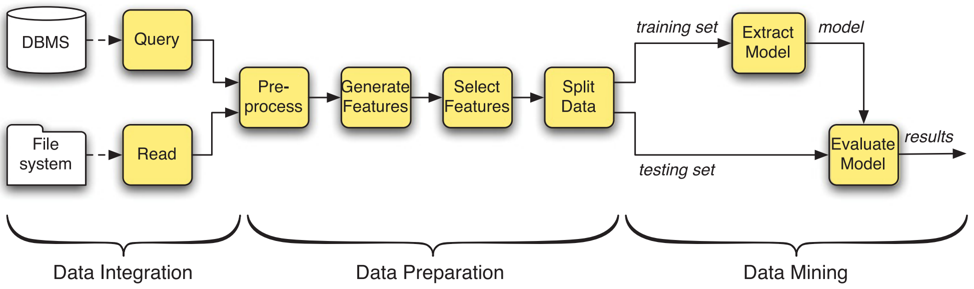

Fig. 1. Common workflow in scientific experiments.

require logical and conceptual notations that enable a wide range of researchers in science, biomedicine, and engineering to take charge of their data-intensive methods.

2. WORKFLOWS

The emergence of computational and data-driven science as the third and fourth paradigms increases the demand for modern technologies. With the help of data analysis experts who master statistical methods or data-mining techniques, domain scientists[21] try to discover new knowledge from simulation, observation, and experimental data. This often involves (1) moving data from data sources to computational resources; (2) cleaning, calibrating, and normalizing data; (3) constructing a model using part of that preprocessed data; (4) validating the model with the remaining data; (5) visualizing the results; and (6) moving the results to a storage system. Simulations explore and test the implications of mathematical models of phenomena; they may be included in workflows as a source of data. For example, for the recent discovery of gravity waves [Abbott et al. 2016], the parameters describing the masses, momentum, and separation of the colliding black holes had to be adjusted until the output from a simulation matched the detected signal—this used the Pegasus workflow system. Similarly, to develop tomographic Earth models, through seismic inversion, the wave propagations from many earthquakes have to be simulated, and the finite-element models of wave velocity have to be adjusted by adjunct wave propagation until the results match the seismic observations [French and Romanowicz 2015]. Such processes can be modeled as workflows, which are defined here as a set of interrelated computational and data-handling tasks designed to achieve a specific goal.

2.1. Workflow Characteristics

A workflow comprises three components: a list of tasks or operations, the set of dependencies between the interconnected tasks (the flow), and the set of data resources used to generate or terminate the flow.[22] In a graph representation, the tasks and data resources are the vertices and the dependencies are the edges connecting vertices, as shown in Figure 1. The edges can represent two kinds of dependency: control flow and data flow [Shields 2007].

Control-flow graphs comprise tasks and precedence constraints. The tasks are *operations* and edges specify the order of operations. An example workflow in Figure 1 demonstrates a basic pattern. The two tasks in the data integration phase may be run

[21]In commerce, the person who grasps the business issues is called the "domain expert," for example, the marketing strategist, financial controller, or logistics planner. In science, an expert in the discipline fills that role.
[22]Data resources include data sources and data sinks (e.g., data stores and archives).

concurrently, while the tasks in the data preparation stage form a sequence that runs after these have both completed, wherein each task is executed before the tasks at the arrow end of connecting edges. The sequence is then split, but the arc from ExtractModel to EvaluateModel ensures a model has been produced before it is evaluated. A workflow graph can be directed cyclic (DCG) or directed acyclic (DAG). The main difference is that DCG supports iteration and DAG does not. Bharathi et al. provide a characterization of scientific workflow structures [Bharathi et al. 2008], and van der Aalst et al. describe basic workflow patterns [van der Aalst et al. 2003]. The Workflow Pattern initiative catalogs more patterns [van der Aalst and ter Hofstede 2014].

In a *data-flow* graph, dependencies between tasks represent flows of data. Data move along arcs and are transformed by tasks. If Figure 1 denotes a data-flow graph, then data from the Query operator (metadata) and data from the Read operator (raw images) flow into the Preprocess operator. The Preprocess operator transforms every data item (raw image) and transmits the results to the succeeding operator Generate Features. Data-flow graphs permit the operators' executions to overlap in a processing pipeline.

These workflow graphs are logical models, known as *abstract workflows*, defining the steps to be taken in scientific experiments. Abstract workflows define the tasks and their dependencies. To run the experiments, the tasks need to be mapped to executable software components, generating a *concrete workflow*. The workflow life cycle has been defined for both business and scientific domains [Deelman et al. 2009; Görlach et al. 2011; Ludäscher et al. 2009], each proposing their own sequence of phases. Görlach et al. [2011] suggest three phases, while Ludäscher et al. [2009] suggest four phases, with a "workflow preparation" phase staging data into computing resources prior to the execution phase. Only Deelman et al. [2009] discuss a "provenance capture" phase, collecting information for workflow reproducibility. However, they all make the following observations, which apply for all forms of data-driven analysis:

—These phases are from the scientists' perspective as they create and run workflows.
—Scientists compose, operate, analyze, and refine workflows.
—Scientific workflows are exploratory; that is, it is common to reuse workflows and refine them using trial and error.
—Scientific methods are often repeated; that is, scientists rerun workflows with different parameters and datasets.
—Runtime monitoring and diagnostics are important; that is, scientists monitor progress and may steer or decide to abort or suspend an execution.

Spinuso has pioneered the active use of provenance at runtime to trigger responses to conditions and to support job, data, and research-campaign management [Spinuso et al. 2016]. Making provenance *immediately* useful is a significant step in engaging researchers [Myers et al. 2015].

2.2. Workflow Architectures

A wide range of WMSs have been developed, for example Pegasus [Deelman et al. 2015], Kepler [Ludäscher et al. 2006], Taverna [Wolstencroft et al. 2013], Triana [Taylor et al. 2007a], Swift [Zhao et al. 2007], Trident [Barga et al. 2008], Galaxy [Blankenberg et al. 2010], ASKALON [Fahringer et al. 2007], WS-PGRADE/gUSE [Kacsuk et al. 2012], Meandre [Llorà et al. 2008], and Apache Airavata [Marru et al. 2011]. Studies [Taylor et al. 2007b; Goble and De Roure 2009; Görlach et al. 2011] identify the following roles for workflows:

—Support for collaborative research by enabling scientific communities to share automated and formalized processes such as data analysis

—Construction free from distracting details about workflow management and execution
—The ability to automate workflow steps, that is, their mapping and execution, and to repeat in silico experiments
—Integrating resources from distributed and heterogeneous enactment platforms
—Handling large volumes of data and complex computations
—Improving the execution through various optimization strategies

We summarize several studies that classify the architecture of WMSs; they propose widely used taxonomies. We then introduce potential improvements.

Becker et al. [2002] identify three classes of business process: (1) workflow-supported *organizational processes* (facilitating human actions during those processes), (2) workflow-driven *software processes* (entirely automated computational tasks), and (3) *hybrid processes* (a mixture of the two). They add an organizational dimension, that is, inter- and intra-organization level. Grefen and Vonk [2006] describe a transactional workflow model and discuss its support from the conceptual (specification language) and the system (workflow architecture) points of view.

Over the last two decades, scientific communities have used workflow technologies to automate their computational experiments that exploit distributed and high-performance computing infrastructures (e.g., grids) and access data, cloud, and HPC resources, which are often geographically dispersed and independently managed. Yu and Buyya [2005] classify various approaches to mapping workflows onto grids. They review 13 existing WMSs and suggest research directions. Deelman et al. [2009] develop a general taxonomy of features for WMSs relevant to scientists and use these to characterize and compare the abilities of WMSs throughout the life cycle.

A number of papers regarding *specific aspects* of WMSs have been published, including the following:

—*Scheduling* — Wieczorek et al. [2009] analyze five facets of workflow scheduling: workflow model, scheduling criteria, scheduling process, resource model, and task model, in each case giving an extensive taxonomy and a survey of related WMSs.
—*Verification and validation* to improve the correctness of grid workflows. Chen and Yang [2008] propose a taxonomy for workflow *verification*: structure, performance, and resources, and *validation* of consistency between processes and specifications.
—*Provenance* — Provenance data are often specific to the WMS that gathers them and prove difficult to integrate across systems. Many of the systems in Section 3 are adopting W3C PROV[23] to enable the interchange of provenance data [Groth et al. 2012]. Da Cruz et al. [2009] distinguish perspectives of provenance (i.e., capture, access, subject, and storage) and provide a taxonomy of provenance. They survey 11 provenance systems, including Pegasus, Taverna, Kepler, and Swift.
—*Data publishing* — Murphy et al. [2015] define data publishing as providing discoverable, standard, and trusted data repositories that allow scientists from different disciplines to access, reuse, and analyze the unique datasets over the longer term. The published data should be well documented, identified, curated, interoperable, and archived. The RDA/WDS Publishing Data Workflows Working Group[24] will analyze a representative range of workflows and standards for data publishing, including deposit and citation, and recommend reference models and implementations for application in new workflows. Garijo et al. [2012] propose the publication of the workflows, components, and datasets as Linked Data [Heath and Bizer 2011] to make scientific workflows more reusable and to increase the reproducibility of scientific

[23]W3C PROV-Overview: www.w3.org/TR/prov-overview.
[24]Research Data Alliance (RDA): www.rd-alliance.org.

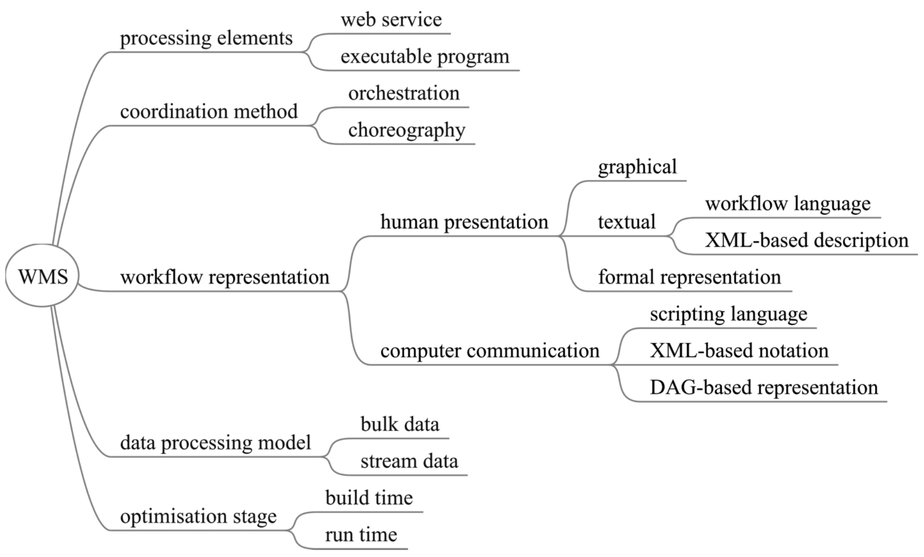

Fig. 2. Architectural characterizations of WMSs.

results [Garijo 2015]. EUDAT[25] and DataONE[26] are each developing a data infrastructure compliant with the Open Archival Information System (OAIS).

These studies improve our understanding of WMSs. However, they omit a few characteristics from their taxonomies, which are illustrated in Figure 2.

Processing elements (PEs), the building blocks of workflows, are *software components* that encapsulate a particular functionality to perform their task. Gannon [2007] distinguishes two types of workflow depending on the way a PE is implemented: (1) *component-based* workflows, also known as *task based,* are accessed through a specific interface, such as a function call, an interprocess communication, or a job submission—they may be written in any language and need to be deployed explicitly during enactment, (2) PEs in *service-based* workflows are implemented as web services, which are self-contained and self-describing programs exposed via web servers [Wang et al. 2004] that are invoked and respond using web service protocols addressed to already-running instances.

The coordination of the execution differs significantly in the two types of workflow. In component-based workflows, the PEs are often stand-alone applications that receive input data, perform their task, and produce a result. A WMS deploys and connects these PEs together by fetching results, often as files, and supplying them as input to subsequent components. In service-based workflows, the web services are independent, predeployed, and potentially dispersed web instances. A workflow is constructed by a collection of web services communicating with each other and with a WMS controller through message passing, with an explicit coordination method.

The *coordination method* falls into two categories: *orchestration* and *choreography*. *Orchestration* has a single controller and oversees the execution flow and invokes services based on a workflow written in an orchestration language, WS-BPEL.[27] Written

[25]EUDAT: www.eudat.eu B2 services.
[26]DataONE: www.dataone.org.
[27]Web Services Business Process Execution Language (WS-BPEL): docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

initially for business applications, BPEL has been adapted to the scientific domains [Emmerich et al. 2005; Slominski 2007].

Choreography describes a collaboration between services to achieve an agreed-upon goal; it involves messages between multiple parties, where no one party truly owns the conversation [Barker et al. 2009]. WS-CDL[28] is used for choreography. Service orchestration works well in business domains, but not for scientific applications, where data and applications are often large and use dispersed services managed by multiple organizations, because service choreography invokes more communication. Barker et al. [2008] propose a hybrid model with centralized control flow but distributed data flow, which provides robustness and reduces data movement.

The *workflow representation* can meet two goals: (1) *human presentation*, an external representation for creators and editors of a workflow, with graphical, textual, and formal variants, and (2) *computer communication*, an internal representation used for communication between subsystems to achieve enactment. The graphical representation may facilitate the composition of workflows using GUI editors. It increases the usability of the WMSs but may not be suitable for describing workflows with a large number of tasks in detail, which leads to textual representations that may be human comprehensible or XML based. The representations denoting abstract workflows used by workflow editors are transformed into internal representations and are passed to the workflow manager to organize the execution. It may apply graph transformation before generating a concrete execution plan to be sent to the execution engine. Three common internal representations are scripting languages, XML-based descriptions, and internal DAG-based representations. These representations are also employed for storing workflows [Elmroth et al. 2010].

The *data processing model* of a workflow matches the internal data processing model of the PEs, which can be divided into *bulk data*, where PEs receive whole datasets for example, files (which may contain multiple data elements), and produce their results as bulk data, and *stream data*, where data units arrive from continuous and time-varying data streams, such as the output from sensors [Babcock et al. 2002]. For stream data, a PE produces data units on its output streams for each data unit that arrives on an input stream. It is best to describe both using an operations/operators terminology. In the bulk data model, PEs are *operations*, which are instantiated to process a dataset and terminated after it has been processed. WMSs execute these operations in sequence, and some operations may be executed concurrently provided there are no interdependencies between them. This is called *batch processing*. For stream data, PEs are *operators* that keep on working on data items as they arrive and will not be terminated as long as more data is expected. These operators can be connected to form a *pipeline*, a successor operator processing the data items its predecessors have produced, their execution therefore overlapping. The choice between batch and stream processing may be determined by nonfunctional requirements, such as continuously monitoring an observed system, or it may be based on the optimization issues discussed later and lead to research questions presented in Section 4.

The last characteristic concerns at which stage which optimization is performed: *build time*—workflow composition and planning the mapping to resources, or *runtime*—the deployment, execution, and monitoring phases. An execution engine uses status information and size of data to dynamically optimize. Build-time optimization focuses mainly on graph transformation, for example, task clustering and parallelization.

These characteristics all concern how WMSs operate in practice. They significantly affect how WMSs *inter*operate, for example, how workflows are expressed, managed, operated, and optimized. Section 3 analyzes current WMSs in terms of those

[28]Web Services Choreography Description Language (WS-CDL): www.w3.org/TR/ws-cdl-10/.

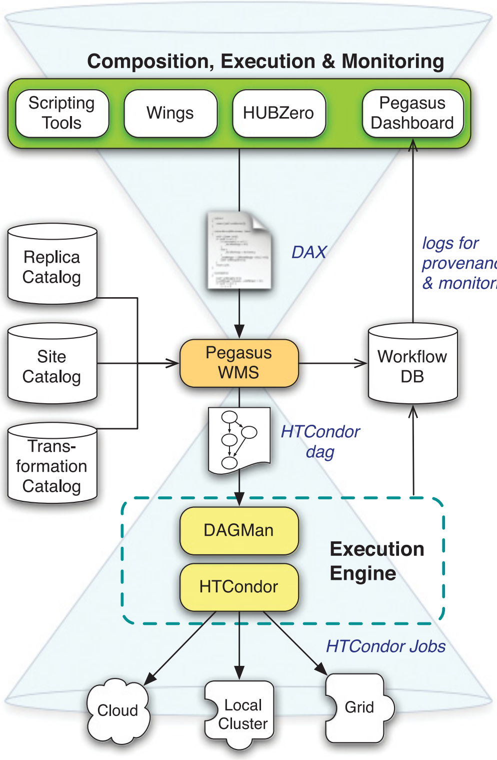

Fig. 3. Pegasus architectural diagram.

characteristics, as researchers will consider them when deciding which *existing* WMSs to adopt. In Section 4, we discuss how the limitations of existing WMSs may be overcome.

3. REVIEW OF SELECTED EXISTING WORKFLOW MANAGEMENT SYSTEMS

A review of all WMSs is not feasible, so we discuss seven: Pegasus, Kepler, Taverna, Swift, KNIME, Airavata, and Meandre. The first six are well established and widely used in multiple domains. Meandre is less widely used, but its data-flow system exploits fine-grained data streaming and is closest to our view of future trends (see Section 4). We briefly review their technology and discuss their salient features, such as architecture, development environment, and workflow language, and conclude this section with a summary based on the additional characteristics defined above.

To aid our comparison of these WMSs, we sketch a system architecture diagram for each one (e.g., see Figure 3) that shows the workflow composition tool (colored in green), the resource mapping mechanism (colored in orange), and the workflow execution engine (colored in yellow). We superimpose that system architecture diagram onto an hourglass figure—the upper part of the hourglass denotes the development and refinement of the logic in the workflow and supports activities such as sharing and debugging; the lower part represents computation across diverse distributed computing infrastructures (DCIs) evaluating instances of the workflows. The arguments for this "hourglass model" are given in Atkinson and Parsons [2013]. Both the upper and lower cones grow as capabilities are added and as the set of available DCIs evolve—see Section 4. The narrow neck of the hourglass allows each part to evolve independently of the other, protecting scientific and user-support investments in the upper cone from obsolescence as DCIs evolve. There is a challenge developing a stable and sufficiently powerful notation that is not too closely tied to a target DCI; for example, Pegasus (Section 3.1) uses DAX at an upper level and HTCondor DAG at a target-specific level, whereas Taverna (Section 3.3) is now on its third version of a language, SCUFL2, to communicate between composition and execution. Wider integration and mappings that bridge technologies are required to meet interdisciplinary challenges (see Section 4.1), making the design of this critical and stable communication channel a key research challenge—most current scientific workflow systems leak properties of the underlying platform into the upper cone, thereby distracting users and causing lock-in.

3.1. Pegasus

Pegasus[29] is a well-known WMS that is widely used across domains, for example, earth science [Maechling et al. 2007] and astronomy [Berriman et al. 2010]. Together with Wings, DAGMan, and HTCondor,[30] it provides a complete workflow solution for handling scientific experiments. Wings is a semantically rich workflow system, used to create and validate workflows and generate metadata. Workflows are created and stored in workflow libraries. At this stage, they are *workflow templates*, logical definitions of process plans, with no bindings to data or executable programs. The metadata that semantically describe the components and requirements of the workflow templates are stored in repositories, so they may be discovered, shared, and reused by different users and experiments. Wings helps researchers find templates and data to create *workflow instances*, which are also known as *abstract workflows*. Workflow instances have the data to be used specified, but are still independent from the execution resources. Pegasus maps the workflow instance onto execution resources to create an *executable workflow*, which is fully specified: the data and their location, the computing resources, and the required data movements. DAGMan and HTCondor take over and execute the workflow on a distributed environment. Figure 3 shows the architecture with interaction between these subsystems.

Wings plays two roles in the life cycle: workflow composition [Gil et al. 2006] (semantically rich construction) and provenance tracking [Kim et al. 2008] (provenance of workflow instances and metadata for data products). The second phase of the workflow life cycle, resource mapping, is handled by Pegasus. Pegasus is a workflow planner and does not execute workflows. It can exploit various execution engines, for example, HTCondor and Globus.[31] Its input is an abstract workflow written in an XML format, called DAX, from which it generates a concrete workflow as the input to DAGMan.

The mapping relies on three catalogs. The *Site Catalog* describes the available compute resources. A site can be a cluster, virtual machines in clouds, or local machines. Pegasus exploits heterogenous DCI spread across grid and cloud [Deelman 2010]. The *Replica Catalog* maintains a mapping from logical to physical file names for data discovery. The *Transformation Catalog* maps logical operations to physical executables. A user can define whether a component is stageable from other sites. Pegasus uses Kickstart [Vöckler et al. 2006] to launch programs and capture their exit status and monitoring information, which are then stored in the Provenance Tracking Catalog [Deelman et al. 2006] and used for debugging.

Pegasus is popular because (1) its planner automatically adds *staging* and *registration* jobs to the concrete workflow; (2) it is flexible and scalable [Callaghan et al. 2010]; and (3) it performs optimization, such as clustering small jobs together, automatically

[29]Pegasus: pegasus.isi.edu [Deelman et al. 2015].
[30]Wings: wings-workflows.org; HTCondor: research.cs.wisc.edu/htcondor/.
[31]Globus: www.globus.org.

releasing storage, and reusing results from previous runs [Chen and Deelman 2011]. With the integration into the HUBZero framework [McLennan and Kennell 2010], Pegasus has extended its powerful workflow automation and management services to a wider research community [McLennan et al. 2015].

A key architectural question concerns the value of a mapper; a WMS could require composition using executable components explicitly. Pegasus demonstrates the benefits of abstraction that separates the workflow design from the target technology. This lets scientists focus on their scientific work without being distracted by low-level details. It increases reusability: allowing the same workflow to apply to different datasets and to run on different DCIs. Abstraction increases the scope for optimization.

3.2. Kepler

Kepler[32] originated from the Science Environment for Ecological Knowledge project,[33] which combined EcoGrid (data storage, sharing, and analysis), Semantic Mediation (reasoning to discover and integrate data), and Analysis and Modeling (visual environment for ecologists to compose workflows—motivating Kepler) [Michener et al. 2005]. Kepler has become a general workflow infrastructure supporting many domains, including chemistry,[34] geology,[35] molecular biology,[36] and oceanography.[37]

Kepler is built on Ptolemy II [Eker et al. 2003], which facilitates actor-oriented computation [Bowers and Ludäscher 2005]. This model matches the exploratory nature of scientific workflows during design, prototyping, and execution. Each process is modeled as an *actor* that encapsulates required functions. Actors are independent and communicate using message passing.

To achieve different execution semantics within a single architecture, Kepler separates the orchestration from the execution engine and uses *directors* to organize collaborating actors. The actors define "what" are the processing tasks and the directors determine "when" their processing occurs. Kepler supports the following coordination models: Process Networks (PNs), Dynamic Dataflow (DDF), Synchronous Dataflow (SDF), Continuous Time (CT), and Discrete Events (DEs). These meet different requirements: CT and DE are used for workflows that depend on time (e.g., processing data from sensors and analyzing population growth); DDF and SDF are used to transform and filter non-time-series data; and PN manages parallel threads and distributed execution.

The actor/director model gives Kepler extensibility and flexibility (see Figure 4). It can be extended easily by developing the necessary set of actors, such as:

—integrating applications, for example, use actors RExpression and MatlabExpression to run an R or a Matlab, respectively;
—integrating web services, for example, WebService as an actor for WSDL and actors for RESTful web services, and Opal[38] that wraps scientific applications as web services;
—data movement with specific protocols, for example, GridFTP, SSHFileCopier, and FTPClient

[32]Kepler Project: www.kepler-project.org [Ludäscher et al. 2006].
[33]Science Environment for Ecological Knowledge: seek.ecoinformatics.org.
[34]RESearch sURGe ENabled by CyberinfrastructurE: ocikbws.uzh.ch/resurgence.
[35]Geosciences Network: www.geongrid.org/.
[36]Scientific Data Management Center: sdm.lbl.gov/sdmcenter/.
[37]Real-time Observatories, Applications, and Data Management Network: www.roadnet.ucsd.edu/.
[38]Opal: nbcr.ucsd.edu/data/docs/opal/.

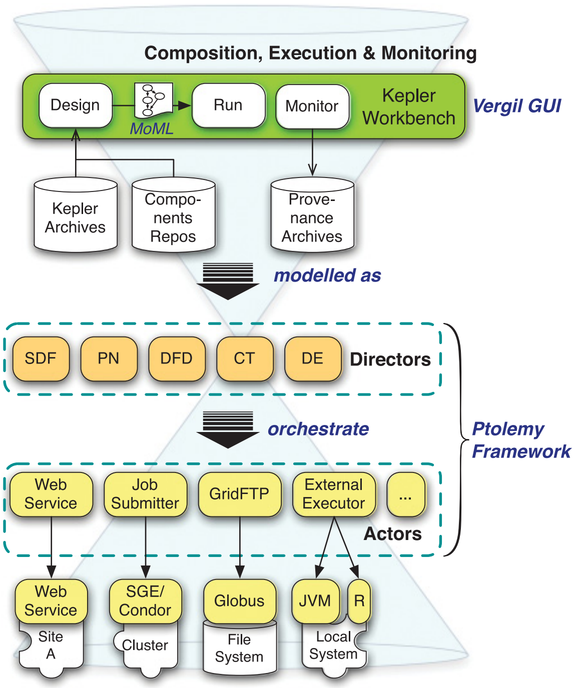

Fig. 4. Kepler architectural diagram.

—interacting with DCI; for example, JobCreator and JobSubmitter create and submit jobs to clusters, GlobusJob submits to Globus, SRBConnect accesses SRB,[39] and DataGridTransfer accesses iRODS services;[40]
—executing shell scripts and applications on local machines using ExternalExecutor.

The set of directors can be extended to support new modes of computation; for example, Abramson et al. [2008] built Nimrod/K on Kepler's runtime engine and created a new Tagged Dataflow Architecture director to achieve dynamic and parallel workflow execution.

Kepler provides high usability through a powerful workbench by using the Ptolemy Vergil GUI [Brooks et al. 2007] that is used to construct and monitor workflows and to access their provenance archives [Altintas et al. 2006; Bowers et al. 2007]. The provenance framework provides APIs for collecting assertions and data dependencies and for querying the provenance database. They deliver provenance-based fault tolerance, such as the Checkpoint actor, "exception handling" that stops a subworkflow when an error is detected [Crawl and Altintas 2008]. Kepler avoids redundant work by using provenance records during recovery [Bowers et al. 2007].

Kepler maintains a searchable repository of actors and workflows to increase their reuse and accelerate workflow development. It has over 350 ready-to-use actors that provide access to the EarthGrid[41] ecological data described using the Ecological Metadata Language.[42] Kepler saves workflows in XML format using Ptolemy's Modeling

[39]Storage Resource Broker (SRB): www.sdsc.edu/srb/.
[40]Integrated Rule-Oriented Data System (iRODS), irods-consortium.org.
[41]Knowledge Network for Biocomplexity: knb.ecoinformatics.org/.
[42]EML: knb.ecoinformatics.org/software/eml/.

Markup Language, which specifies components and parameters. They may be saved in Kepler Archive Format to extend reproducibility as they can then be imported and rerun. Kepler promotes its "smart rerun" mechanism for handling parameter sweeps, where data dependency is used to only execute subworkflows affected by the parameter changes. These features make Kepler a highly usable and automated WMS.

The Kepler provenance work [Cuevas-Vicenttín et al. 2012] has been extended in the DataONE project,[43] a worldwide collaboration to provide a cyber-infrastructure for environmental science. The DataONE Scientific Workflows and Provenance Working Group is developing a provenance architecture for WMSs [Missier et al. 2012], which includes a provenance data model (D-OPM) and query language for D-OPM. They have "stitched together" traces from Kepler and Taverna workflows [Missier et al. 2010a].

3.3. Taverna

Taverna[44] is an open-source, service-based, and domain-independent WMS created by the myGrid team,[45] which has focused on supporting the life sciences community (biology, chemistry, and medical imaging) [Oinn et al. 2006]. myGrid provides tools to help e-Science researchers: (1) Taverna, their workflow management tool; (2) myExperiment,[46] their workflow collaboration facility; (3) SysMO-DB,[47] their data sharing facility; (4) Utopia,[48] their protein sequence and structure analysis tools; (5) BioCatalogue,[49] a curated catalog of Life Science Web Services; and (6) BioVeL,[50] a virtual e-laboratory for biodiversity researchers. This sustained collaboration with the life science community makes Taverna one of the most popular systems for "in silico" experiments.

Taverna makes it easy for domain experts to create workflows via the Taverna workbench. They can obtain workflows from a local repository, a remote URL, or myExperiment—a virtual research environment for sharing workflows [De Roure et al. 2009]. Workflows are data-flow objects serialized as t2flow files. Reusability is achieved in two ways: (1) workflows may be reused with different parameters or datasets, and (2) workflow fragments may be reused when constructing new workflows.

Taverna's libraries offer over 3,500 services, and users can add and announce new services. The service discovery mechanism searches public registries (e.g., UDDI[51] and Grimoires[52]). Services may be specified as URLs or be stored locally [Oinn et al. 2007]. Taverna has built-in services for basic operations (e.g., file I/O).

The Taverna workbench submits workflows to a local or remote Taverna Engine, where instances (i.e., WorkflowInstanceFacade) are created to represent the running workflows, as shown in Figure 5. Two differences distinguish Taverna from Pegasus and Swift:

—Taverna workflows connect web services, coordinate their executions, and arrange for data to flow between them, whereas in Pegasus and Swift workflows denote a logically ordered sequence of computing tasks, which are typically performed by submitting jobs, supplying their data, and collecting their results.

[43]Data Observation Network for Earth (DataONE): www.dataone.org.
[44]Taverna: www.taverna.org.uk/ [Wolstencroft et al. 2013].
[45]myGrid: www.mygrid.org.uk/.
[46]myExperiment: www.myexperiment.org/.
[47]SysMO-DB: www.sysmo-db.org/.
[48]Utopia: utopia.cs.man.ac.uk/utopia/.
[49]NioCatalogue: www.biocatalogue.org/.
[50]Biodiversity Virtual e-Laboratory www.biovel.eu/.
[51]Universal Description, Discovery, and Integration (UDDI) Standard: uddi.xml.org/uddi-org.
[52]Grimoires: twiki.grimoires.org/bin/view/Grimoires.

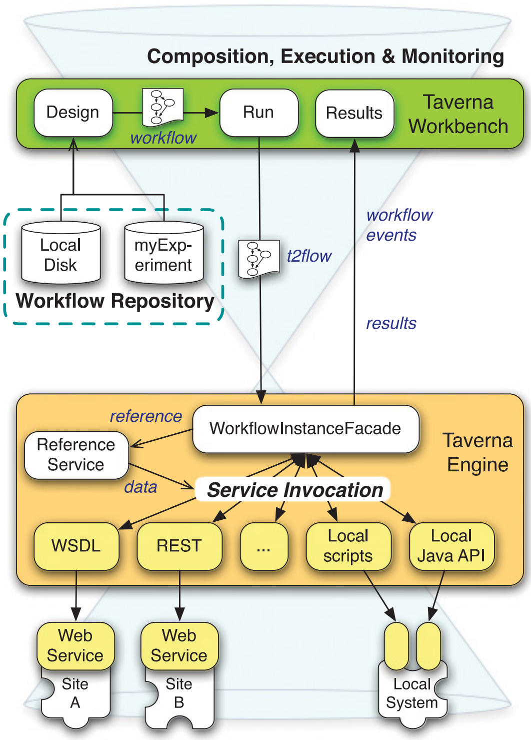

Fig. 5. Taverna architectural diagram.

—Taverna has no centralized enactment engine, the workflow itself performs the enactment (each PE is mapped to an object, which starts its own execution when all of its inputs are ready, and sends its outputs to its successor objects [Missier et al. 2010b]); Pegasus and Swift organize the staging of input data and results and dispatch jobs to their execution platforms.

Taverna invokes the relevant services, sending them references to the actual data. The services then use reference services to retrieve the data. Provenance information is captured for two purposes: execution monitoring (i.e., users can view intermediate results) and reproducibility (i.e., they can reapply a workflow for performance assessment, debugging, or data validation). They do this via the workbench.

Similar to Kepler's actor model, Taverna's plugin model is extensible. Plugins allow Taverna to use more than web services. For instance, the BioCatalogue plugin supports browsing and use of its life science services. The UNICORE plugin enables the use of UNCORE,[53] while the PBS plugin allows submission to PBS queues.

Taverna's workflow language has evolved. In the earlier versions, workflows were written in the Simple Conceptual Unified Flow Language (SCUFL), a high-level XML-based language [Oinn et al. 2004]. SCUFL is a data-flow language that defines a graph of data interactions between web services. However, SCUFL does not have a unified way to extend service definitions via plugins or support for new features in the Taverna Engine. Thus, it was replaced by t2flows, a serializable XML format (easy to be shared and transported) in Taverna 2, which is more verbose but allows finer-grained detail.

[53]Uniform Interface to Computing Resources: www.unicore.eu/.

The SCUFL2 language used in Taverna 3[54] combines the simplicity of SCUFL and expressiveness of t2flows.

Taverna provides a web-based interface, namely, Taverna Player,[55] to allow users to execute existing workflows using a browser. This has eliminated the hassle of downloading and installing local software components [Mathew et al. 2014]. Taverna 2 has been integrated with Galaxy, another popular web-based WMS, in Tavaxy [Abouelhoda et al. 2012], which provides a fine-grained integration of Taverna and Galaxy workflows. Both JSON objects of Galaxy workflows and SCUFL/t2flow of Taverna workflows are translated into tSCUFL objects in Tavaxy, to allow design-time integration. Users can now include Taverna workflows as part of their Galaxy workflows.

3.4. Swift

Swift[56] was initiated by the GriPhyN project to automate the processing of large datasets from high-energy physics experiments. From a simple virtual data language, Swift has matured into a powerful parallel scripting language [Zhao et al. 2007] with an extensive runtime system based on CoG Karajan [von Laszewski and Hategan 2005] that efficiently runs large-scale loosely coupled computations on clusters, clouds, and grid resources for different domains, for example, medical research [Stef-Praun et al. 2007], protein structure modeling [Adhikari et al. 2012], and climate modeling [Woitaszek et al. 2011]. The Swift scripting language, SwiftScript, provides *data-oriented constructs* to specify processing of collections of files by mapping file-system objects into Swift variables with iteration and branching over them. Swift automatically parallelizes processing, chooses computing sites, handles staging of input and output files (specified by mappers), and invokes remote execution. It formalizes and abstracts applications as functions, with input files as parameters and output files as results.

Figure 6 shows the Swift architecture with a set of services to deliver parallel, distributed, and efficient workflow execution. A SwiftScript can be constructed using any editor; the SwiftScript compiler then produces an abstract computation plan. This is dispatched to execution sites, described in the *site catalog*, by the execution engine, CoG Karajan, to obtain remote job execution, file transfer, and data management through abstract interfaces called *providers*. A *data provider* supports file transfer and data management via a wide range of protocols, for example, GridFTP, SCP, FTP, and direct copy. An *execution provider* enables the job execution via a variety of schedulers, for example, GRAM, HTCondor, Sun Grid Engine, and Portable Batch System. The provider interfaces allow Swift to be easily extended to other DCI environments. Swift supports task execution using a provisioning and dispatching system, for example, Coasters [Hategan et al. 2011] and Falkon [Raicu et al. 2007]. Coasters is a node provisioning system for DCI that supports *pilot jobs*[57] on grid, cluster, and cloud resources.

The Swift execution model is simple: noncollection data elements are single assignment and the functional formalization enables implicit parallelization. The foreach construct specifies that the functions applied to the elements defined by its in clause may be executed in parallel. The evaluation of the Swift script is centralized and may become a scalability bottleneck. Then Turbine [Wozniak et al. 2013a], a distributed data-flow engine, can be used as the Swift runtime [Wozniak et al. 2013b].

[54]Taverna is moving to the Apache Incubator: taverna.incubator.apache.org.

[55]Taverna Player: mygrid.github.io/taverna-player/.

[56]Swift: swift-lang.org [Zhao et al. 2007].

[57]Pilot jobs make a series of jobs appear as one job to the host system. They are distributed to remote sites and signal to a scheduler when they are ready for another job. This avoids repeated queuing and setup.

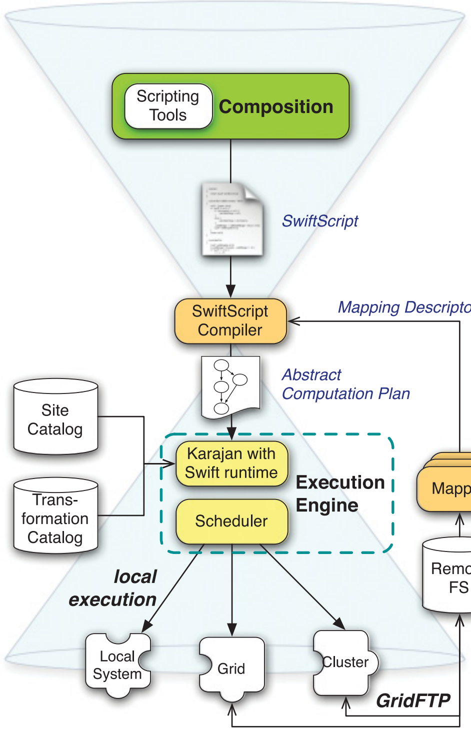

Fig. 6. Swift architectural diagram.

Like Pegasus, Swift use the VDS Kickstart to record provenance. Swift replicates and automatically resubmits failed invocations. It does not provide a workbench for workflow composition but is used as the back end for Science Gateways [Wu et al. 2010] and for Generic Portals for Science Infrastructure [Uram et al. 2011]. Swift enables Galaxy to run large-scale workflows on parallel DCI [Maheshwari et al. 2013].

3.5. KNIME

The KNIME[58] Desktop is an open-source, workflow-based, data analysis platform [Berthold et al. 2009]. Its GUI is an Eclipse workbench, with panes for designing workflows, listing workflow components (called nodes), describing nodes, organizing workflows and projects, viewing execution error messages, obtaining workflows from servers, and so on. KNIME is used for applications including social media analysis, game analytics, pharmaceutical research, and chemo-informatics [Beisken et al. 2013].

A KNIME workflow is created by dragging nodes onto the design pane and then configuring and connecting these nodes using the workbench (see Figure 7). Configuring a node sets its parameters, such as specifying the file name, path, and column delimiters for a file reader node, or the number of clusters and maximum number of iterations for a k-means clustering node. Connections between nodes transfer data (for processing and analyses, or to configure the subsequent nodes) and can transfer models (such as a derived classification tree).

A workflow can be executed entirely or up to a selected node. Partial execution of a workflow aids debugging by allowing a user to inspect intermediate data and models, to reset node parameters, and to rerun the workflow or resume its execution from a

[58]KNIME: www.knime.org [Berthold et al. 2009].

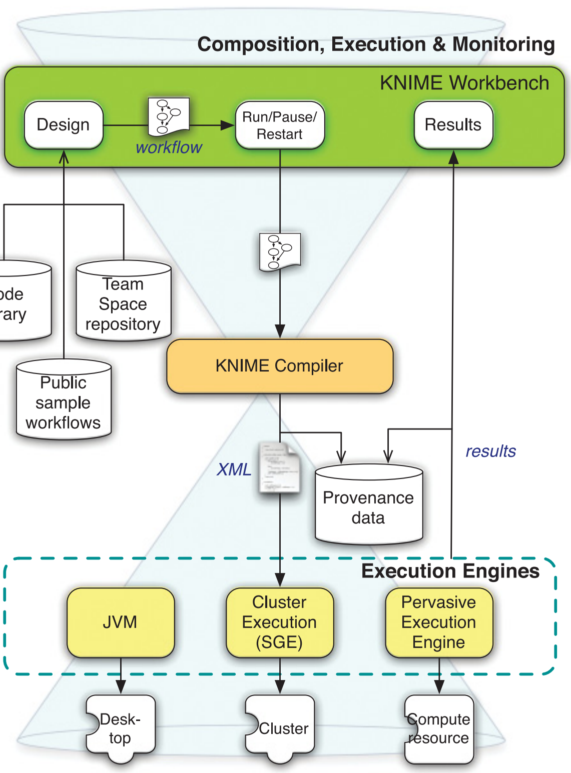

Fig. 7. KNIME architectural diagram.

checkpoint. A workflow is stored in its project directory, which stores all workflow nodes and their settings (in XML), and any data and models that they produce.

The repository provides an extensive library of components with node categories that include I/O, database, data manipulation, mining, and flow control (i.e., loops, switches, and variables). This is supplemented by integration with packages such as R and Weka, and user-community contributions in several application areas including image processing, bio- and chemoinformatics, and text retrieval. Comprehensive documentation is provided for users developing their own KNIME nodes (by extending specified Java classes and creating an XML file describing the node and its configuration options). The KNIME Desktop also provides access to public workflows for reuse.

The open-source KNIME Desktop provides opportunities for collaboration via the import or export of KNIME workflows. It also supports optimization of large or compute-intensive workflows insomuch as the nodes automatically exploit multithreading and may be set to cache data to disk to improve throughput. These requirements are catered for more effectively in some of the KNIME commercial extensions: KNIME Team Space allows users to work within a shared space, KNIME Cluster Execution enables workflows to be executed on a cluster, while KNIME.com's partnership with Pervasive[59] has resulted in RushAnalytics for KNIME, giving access to Pervasive's execution engine that uses horizontal, vertical, and pipelining parallelism.

[59]Pervasive: bigdata.pervasive.com.

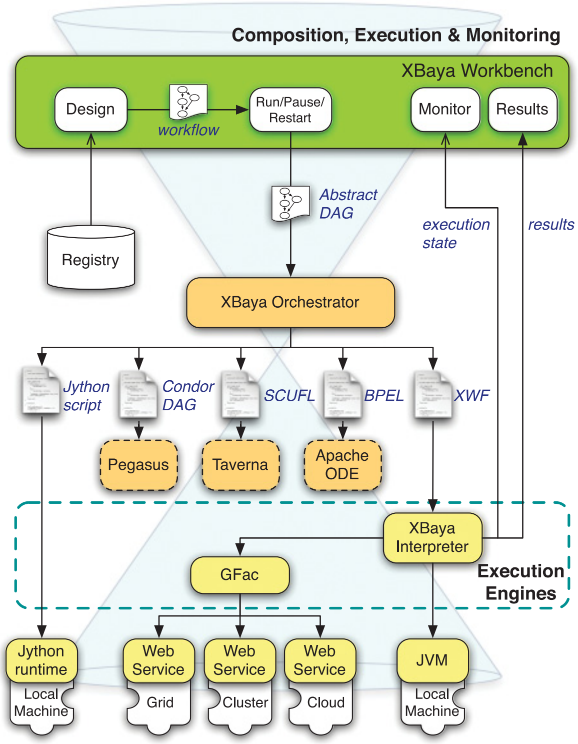

Fig. 8. Airavata architectural diagram.

3.6. Apache Airavata

Airavata[60] is an open-source campaign developing a WMS that executes applications on a variety of DCIs. The architecture (see Figure 8) is service oriented and use distributed messaging for workflow composition and orchestration. It includes XBaya, GFac, and Registry-API, and thereby provides a complete workflow solution for in silico experiments. Airavata is used in various projects, for which it delivers (1) a dynamic workbench to execute workflows on Amazon EC2 for BioVLAB [Yang et al. 2010], (2) computational workflows for SEAGRID/GridChem [Dooley et al. 2006], (3) web services and workflow orchestration for oreChem [Challa et al. 2010], and (4) parameter optimization and analysis for ParamChem [Ghosh et al. 2011].

XBaya is a workflow suite for Airavata. It consists of a GUI that is used for workflow composition and monitoring. Users create workflows via the XBaya workbench using a drag-and-drop GUI. An abstract DAG that is independent of target platforms is then generated. The resulting workflow can be mapped to various targets, such as BPEL 2.0 for Apache ODE [Gunarathne et al. 2009], SCUFL for Taverna, DAGman for Pegasus, Jython scripts, and Java. Users can select the workflow runtime that suits their applications.

XBaya has an interpreter for dynamic and interactive workflow execution. When users launch a workflow, the interpreter starts executing the workflow DAG. As in the case of KNIME, Airavata demonstrates the benefits of an execution model where users can stop and resume the execution of workflow at any time as the interpreter provides

[60]Apache Airavata: airavata.apache.org [Marru et al. 2011].

fine-grained control. This enables users to reconfigure an active workflow and resume its execution with the update immediately incorporated by the interpreter.

When users have submitted a workflow for execution, they can monitor its progress using the XBaya monitoring component, which provides the state of job submissions to batch queues and the progress of data transfers. This component supports synchronous, when the workflow uses the interpreter, and asynchronous, when the workflow has been submitted to a batch queue, that is, monitoring. Airavata uses RabbitMQ[61] [Marru et al. 2015] to send WS-Event notifications between XBaya, the workflow interpreter, monitoring, and GFac. Visualizations of WS-events let users observe their workflow's progress—similar to active provenance in Spinuso et al. [2016].

The Generic Application Service GFac provides a framework to wrap an application in a service interface. GFac can generate SOAP, REST, and Java interfaces to applications. Application providers register their applications by providing definitions of inputs, outputs, work-space directories, and remote access mechanisms. Once applications are registered, GFac constructs requests to computational resources to host specific operations with support for file staging and security.

Airavata has a thick-client API to achieve portability across infrastructures. The registry API can be reused by XBaya and GFac to store and retrieve data. This provides a unified API that can be layered on top of various content repositories. In addition, the API is used to catalog workflow inputs and outputs.

Using their browser, users can experiment with workflows and retrieve their output using the Science gateway.[62] This eliminates the complexities of installing Airavata software but assumes sufficient computational resources behind the gateway. Application developers register workflows. Users initiate runs by supplying data and parameters to those workflows.

3.7. Meandre

Meandre[63] is a semantically enabled, web-driven, data-intensive, flow execution environment developed under the Software Environment for the Advancement of Scholarly Research[64] project, which created a virtual research environment for humanities scholars to exploit the rich digital data becoming available in their disciplines and to share their data and research. The design principles of Meandre aim for a robust and scalable system for data-intensive research, scaled from a single laptop to a high-performance cluster, with collaboration encouraged by sharing components [Llorà et al. 2008]. Web-scale music analysis using NEMA[65] used Meandre to run genre classification workflows on the NCSA[66] supercomputing facility [De Roure et al. 2011].

Among the WMSs in this section, Meandre is the only one using a data-streaming model. It has two types of *component* connected to form a *flow*—a workflow in our context: (1) *executable components* that perform computational tasks without human interactions and are executed according to their predefined firing policy and (2) *control components* that permit user interaction via an HTML form or an Applet.

Meandre's approach to fostering sharing and increasing reusability of components and flows uses semantic web RDF metadata to cross application domain, enterprise, and community boundaries. Metadata for components and flows have the form name, description, tags, and right. Executable components have additional metadata describing

[61]RabbitMQ: www.rabbitmq.com.
[62]Apache Airavata web gateway: testdrive.airavata.org/.
[63]Meandre: seasr.org/meandre/ [Llorà et al. 2008].
[64]Software Environment for the Advancement of Scholarly Research (SEASR): seasr.org/.
[65]Networked Environment for Music Analysis: www.music-ir.org/?q=nema/overview.
[66]NCSA: www.ncsa.illinois.edu/.

behavior and location, for example, firing policy, runnable, format, and resource_location. Flow components have additional metadata describing connections, for example, component instances, connectors, connector instance source, and connector instance target. The RDF metadata are interpreted by the execution engine to find and initialize components, to determine the form of connection between them and when to execute them.

Meandre has three parts [Ács et al. 2010]: (1) tools for creating components and flows, the Meandre workbench, and an Eclipse[67] plugin; (2) a high-level workflow language, ZigZag; and (3) a semantically described service-oriented execution environment. The Meandre workbench offers discovery, creation, and execution of flows by dragging and dropping components from the repository panel and linking them by clicking on their ports. A declarative language, ZigZag, defines flows as directed graphs.

Meandre has a compiler to convert ZigZag flows into self-contained tasks, called Meandre Archive Units, as files with a .mau extension containing metadata describing the components and flows, and their implementation. Their heterogeneity and scalability are hidden from users. The mau file can be executed by the Meandre execution engine on a laptop or as a batch job on a grid environment via SGE. These files can be shared via myExperiment [De Roure et al. 2010]. A significant achievement is automatic parallelization, that is, the [+AUTO] tag tells the compiler it may parallelize an instance, or users can specify parallel instantiation (e.g., [+4] creates four instances).

Meandre offers a simple and flexible environment for data flow; its server has a metadata store, user-interaction services, and an execution engine. Scalability is achieved by the server being instantiated on demand, that is, run as a single server locally or as multiple servers on a cluster (as shown in Figure 9). The Meandre server can be managed through a web GUI, where users can browse and manage their shared components and flows, and run and monitor flows. The engine initiates a thread for each component and executes them based on their firing policy, recovering resources after termination.

3.8. Summary

We revisit the architectural characterisations from Section 2.2, summarized in Table I.

Pegasus and Swift have similarities because they evolved from the GriPhyN VDS project; their processing elements are executable programs. They have a logical workflow layer that encodes data flow between and temporal ordering of their processing elements. They use catalogs to map logical names to physical files and programs. Both handle large workflows; examples involving more than a million tasks have been reported. Pegasus has the *bulk data* processing model (Section 2.2) with work underway to support the datastream model. Swift uses pipeline execution to improve efficiency [Zhao et al. 2007].

Kepler, Taverna, and Airavata originated separately with different communities (i.e., ecology, life sciences, and meteorology), but they had the same raison d'être: to facilitate scientific experiments using web services and data integration across organizational and geographical boundaries. They all provide an easy-to-use workbench to enable scientists to design and run their workflows, and support program execution on local machines and web services. Kepler uses pipeline execution [McPhillips and Bowers 2005] and has run streaming workflows on cloud platforms [Dou et al. 2011; Zinn et al. 2011; Kohler et al. 2012]. Taverna uses pipelined streaming to reduce workflow execution times [Missier et al. 2010b]. Their data granularity differs, however: Taverna performs a coarse-grained pipelining by allocating a thread for each input

[67]Eclipse: www.eclipse.org/.

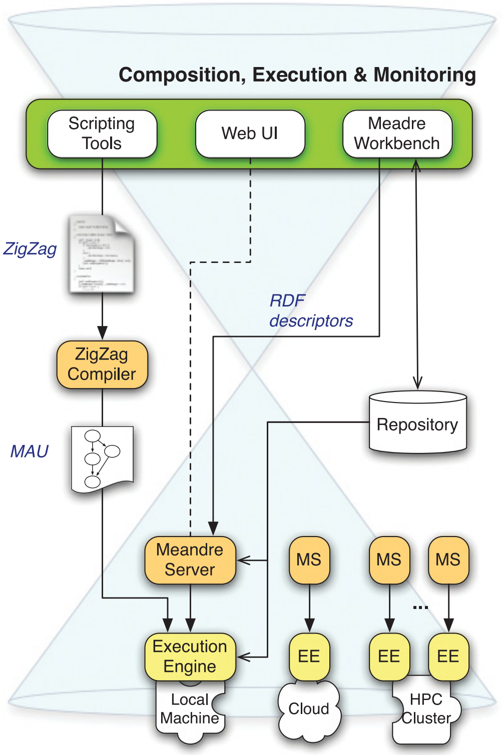

Fig. 9. Meandre architectural diagram.

Table I. Characterizing the Workflow Management Systems

	Pegasus	Kepler	Taverna	Swift	KNIME	Airavata	Meandre
processing element	executable program	executable program & web service	executable program & web service	executable program	executable program	executable program & web service	executable programs
system architecture	orchestrate	orchestrate	orchestrate	orchestrate	orchestrate	orchestrate	orchestrate
optimization stage	build time	none	none	runtime	runtime	none	runtime
user interface	textual	graphical	both	textual	graphical	both	both
data processing model	bulk data	bulk data & stream data	bulk data & stream data	bulk data & stream data	bulk data	bulk data & stream data	stream data

item, while Kepler supports pipelining of nested collections [McPhillips and Bowers 2005]. Neither performs fine-grained data streaming.

The KNIME commercial extension, that is, Cluster Execution, supports bulk data processing on a cluster. Both Airavata and KNIME provide fine-grained control over the workflow execution where users can stop, update workflow activity, and resume the execution. Meandre uses web-oriented, data-driven concepts with a streaming data model. Its components are executable programs that process a stream of data. Data analysis experts develop components and publish them in a repository. The workbench is used to build a workflow as a graph of these components.

All seven systems coordinate using orchestration, with a controller overseeing potentially distributed execution. They use a bottom-up approach for workflow construction, where their visual tools or workflow language are used to compose a graph of processing elements, most of which have been previously defined by experts. The level of abstraction in the workflow language varies significantly.

Pegasus does not have a user-oriented workflow language. Its DAX format describes the directed graph that forms the workflow. It is translated to an abstract workflow using two catalogs. DAX requires too much technical information for scientists and changes in the platform require modification of the DAX. In contrast, Swift's scripting language has better abstraction delivered by the SwiftScript compiler and mapper. Its compilation into parallel execution programs is transparent to its users. The mapper reduces explicit data management for large-scale analyses of distributed and heterogenous data. Most scientists find that abstraction and automation improve their productivity. However, in some cases (e.g., for the gravity-wave detection), they require one to inspect every detail to verify the precise validity of the encoding.

Kepler and Taverna have their own workflow languages, that is, MoML and SCUFL. MoML describes workflows rather than abstracting over them. However, Kepler has its own mechanism to hide the complexity and diversity. Its actor/director model is extensible and allows data-intensive engineers to encode powerful patterns. For instance, Kepler has been extended with a tagged data-flow architecture [Abramson et al. 2008].

Users compose workflows in Airavata by building an abstract DAG, which is independent of target technologies. The composed workflow can be mapped to multiple targets (e.g., BPEL, SCUFL, DAGman, Jython, and Java). Using XBaya interpreter or GFac, the workflow can be executed locally or remotely using different execution engines.

The last characteristic relates to workflow optimization. Kepler, Taverna, and Airavata depend on manual optimization, helped by their good provenance and monitoring systems, which provide crucial data for optimization experts. Swift has implicit parallelization and pipeline execution delivered by runtime optimization. KNIME performs runtime optimization that uses multithreading. Meandre has automatic parallelization that multiply instantiates components tagged in the ZigZag script. At build time, Pegasus refines execution plans by (1) workflow reduction (reusing available intermediate data products and removing the corresponding tasks), (2) task clustering (reducing scheduling overhead by submitting groups of small tasks as a composite task) [Chen et al. 2015], and (3) data cleanup (removing data that are no longer needed to release resources sooner) [Srinivasan et al. 2014].

Exploiting today's wealth of data exposes further issues, which trigger research and lead to new workflow execution strategies on sophisticated data-intensive platforms. Those platforms deliver reliable and high-throughput enactment, as they handle scalability, recovery after partial failures, and optimized mappings to the rapidly evolving commercial systems. This delegates much of the responsibility for the lower half of the hourglass to others and amortizes its R&D and operation to a much wider community. The main focus for workflow research, described in the next section, then becomes improvements in the characteristics of the upper part of the hourglass, for example, accommodating greater scale and diversity, delivering better tooling to more practitioner roles, and developing improved mappings to the rapidly evolving data-intensive platforms.

4. SCIENTIFIC WORKFLOW DIVERSITY, SCOPE, AND FLEXIBILITY

In virtually every research domain, the quantity and diversity of data are growing rapidly because the capacity of storage is increasing [Walter 2005], digital communication is pervasive and increases in capacity [Zhao et al. 2011], and the sensitivity, speed, diversity, and deployed numbers of digital data collection devices exhibit a compound growth. This is combined with a growing drive to share data [Interagency Working Group on Digital Data 2009; EU Parliament 2007], enabled by many organizations' standardization efforts (e.g., W3C, OGC, FDSN[68] IVOA, and RDA) and a growing need to combine data across discipline boundaries to address today's societal challenges.[69] This growth in scale and complexity makes automation of scientific methods essential. More and more sciences and researchers will choose workflows to automate and formalize their scientific methods. The benefits include (1) increased productivity and lower error rates as tedious chores are automated, (2) improved scientific methods as many different specialists pool advances to their parts of a method, and (3) achievement of new goals by combining computational power with the increased wealth of data. We review two questions: (1) Why are scientific workflow systems unable to support this increased use? (2) What research will be needed to make them ready?

4.1. Boundaries Limit Growth

Each community develops its own culture—a body of knowledge, established methods, practices, and ethics—shaped for its own research goals and professional practices. It is promulgated to new practitioners through education and induction. It takes effort and leadership for this to incorporate the technological advances and growing data wealth. Inevitably, differences develop, but the foundations of cognate subjects and common factors in the technical environments and working practices provide overlaps that can lead to successful interdisciplinary collaboration, particularly in long-running research campaigns. A crucial form of such interdisciplinary collaboration is synergy between three groups of experts: (1) *domain experts* who identify the key goals and bring scientific insights; (2) *data scientists*: mathematicians, statisticians, and algorithmic experts who formulate steps, simulation of models, and extraction of evidence; and (3) *data-intensive experts* who develop improved ways of mapping methods onto computing infrastructure exploiting technical advances and changes in the forms of provision [Atkinson and Parsons 2013]. Workflow systems provide a framework for this key relationship, but they need to ensure that (1) each group of experts can work effectively (i.e., the higher levels of abstraction in WMSs must meet the needs of the other experts) and (2) the representations available facilitate communication across these key boundaries.

Subdisciplines, organizations, and communities often commit to different workflow systems (e.g., those in Section 3) for many reasons. They develop significant intellectual, cultural, and financial investment in their chosen system, as can be judged by the numbers of components and workflows in their repositories (see Section 3.3). This may build substantial momentum and form identities that have significant value to each community. But it inhibits collaboration, as each technology separates its adherents from similar researchers using a different technology. They use a different language to express their scientific methods, draw on different libraries of components, and depend on different enactment systems. When this happens within a discipline, it means multiple implementations of the key workflows, subworkflows, and components. This may be beneficial competition, but more often, it means effort is wasted, the results are not as easily compared, and improvements created in one technological island do not propagate to the others. Even widely different research domains need very similar workflow fragments; for example, many require a mechanism to enable a researcher to identify a collection of results as valuable and to send them for curation with the issue

[68]W3C: www.w3.org; OGC: www.opengeospatial.org; and International FDSN: www.fdsn.org.
[69]Societal challenges: ec.europa.eu/programmes/horizon2020/en/h2020-section/societal-challenges.

of a persistent identifier (PID) and permanent links to metadata. The organization of such common subtasks can be shared by many disciplines, as in the EUDAT[70] project.

There are already several research campaigns underway to reduce the isolating effect of these technological islands. Some of these are bridges between pairs of workflow systems; examples were given in Section 3. Here we review a sample of more generic approaches. For the upper hourglass, the myExperiment project initiated pooling of workflow repositories across multiple workflow systems [De Roure et al. 2009], enabling workflow developers to search for useful input from any of the participating technological communities. Extended research objects were used to bundle background material with workflow fragments to increase appropriate reuse and share insights. The Wf4Ever project[71] further developed this approach and extended the workflow representation to prolong workflow reuse [Belhajjame et al. 2015]. For the lower hourglass, the project ER-flow[72] enabled the composition of workflows encoded for different systems into a larger "meta-workflow," hiding much of the necessary housekeeping and interfacing from its users [Kacsuk et al. 2014; Terstyánszky et al. 2014]—it accommodates 15 WMSs and maps them to multiple DCIs [Kozlovszky et al. 2014]. However, it leaves the users working with the concepts, terms, and notations of each WMS. To address these conceptual difficulties, Gesing et al. propose integration in the upper hourglass as a meta-workflow composition framework [Gesing et al. 2014].

4.2. Empowering Scientists

Although a great deal of work is routine—using established methods—scientists also need to explore new ways of using data and simulations, to improve existing methods, and to develop understanding of what may be becoming possible. The routine work is well supported by workflows that are packaged via portals in tailored science gateways [Kacsuk 2014].[73] Here experts can invest time in formulating and hand-optimizing the relevant workflows and in coupling them to the portal as this effort is amortized over many repeated uses of a stable method. There remain four problems: (1) as the encoded scientific method is hidden and often includes many technical details, it is no longer reviewed by the scientists; (2) as scientists do not engage directly in the formalization and automation, they do not develop intuitions about what may be becoming possible; (3) as the optimizations are tuned to contemporary technology, the encoding tends to become obsolescent; and (4) as the number and scope of automated scientific methods increase, the shortage of relevant experts to package methods in a science gateway delays advances. The problems with reviewing and understanding these encapsulated workflows are ameliorated by provenance systems that allow their end-users to examine the background to results and trees of derivatives and to request replays [Kim et al. 2008; Cuevas-Vicenttín et al. 2012; Santana-Perez et al. 2016; Spinuso et al. 2016]. The extension of reusability draws on formalizations mentioned earlier [Belhajjame et al. 2015]. Replay is facilitated by bundles that pack input data and parameters with the workflow activation request [Rogers et al. 2013], but bundling data and keeping copies become infeasible as volumes increase and when continuous streams are handled.

To overcome the distancing from workflows, due to gateway packaging and reformulation by experts, it is necessary to keep innovative domain experts engaged with all phases of workflow refinement. Production experience stimulates revision of scientific methods. For most practitioners and for most of the time for innovators, the productivity benefits win over direct engagement. When innovation and quality checks are

[70]EUDAT: www.eudat.eu/.
[71]Wf4Ever: www.wf4ever-project.org/.
[72]ER-flow: www.erflow.eu/.
[73]http://sciencegateways.org/ with relevant publications at http://iwsg-life.org/site/iwsglife/publications.

needed, it is best if domain experts still have a comprehensible and accurate view of the workflow, so that they can explore potential improvements, conducting in silico experiments in a good emulation of the production, up to reasonable scales, depending on automated mappings and optimizations. This is complementary to the input of workflow and DCI experts, who take over the revised workflow and tune it for production before deploying it in a repackaged form. This duality of viewpoints corresponds to the upper and lower hourglass and requires (1) a suitable representation for domain experts to work with that is not obscured by too much detail; (2) automated handling for exploratory and one-off work with the semantics matching that of production exactly for local tests and medium-scale experiments; (3) a means for the other experts to apply their expertise, which will include technical and mathematical detail; (4) extraction of the domain view whenever required; (5) reuse of as much of the expert annotation as possible when revised workflows move into production. The overall effect should be fluent interchange between domain-led method refinement and production running. However well developed automated planners, mappers, and optimizers become, there will remain extremely demanding data-driven science methods where expert statistical, algorithmic, or engineering refinements will be necessary. Research into WMS engineering should reduce the *proportion* of times that this is necessary, but the growth in data and data-driven science will mean the absolute demand will increase. Thus, good tooling for these experts is also an imperative; demand will outgrow their capacity unless their productivity is also improved.

The Wings composition system for Pegasus, SwiftScript, and Meandre's ZigZag (see Section 3.1, Section 3.4, and Section 3.7) provide conceptual models during workflow composition, but the reverse mappings (e.g., for diagnostics) are not supported. The Dispel language focuses on this conceptual level [Martin and Yaikhom 2013], and Atkinson [2013] reports its focus on the logic of data-intensive methods. Meandre (see Section 3.7) aspires to deliver continuity from a method on a laptop to its enactment on a powerful DCI. Virtually all of the workflow systems provide an effective workbench for initial development; KNIME (Section 3.5) and Galaxy [Blankenberg et al. 2010] are particularly successful at delivering comprehensible representations. These representations are typically graphical, but that does not always match the scientists' preferences. For example, the seismologists, like many scientists, prefer to work in the productive environment of tools and libraries provided by Python [Koepke 2014]. As a consequence, it was necessary to wrap the Dispel conceptual model as a Python library, dispel4py [Filguiera et al. 2016], that behind the scenes maps automatically to multiple DCIs to provide the required continuity between development and production. This context illustrates an additional boundary-crossing challenge; the data used for research is collected by long-running observational networks that include aspects of the data preparation—the same is true of astronomical sky surveys and many other shared research infrastructures. The innovative researchers may want to revisit these early stages or may want to propose improvements to them. However, in many of today's research infrastructures, the use of different technologies for data capture and for data-driven research inhibits quick explorations. Service roles, such as hazard monitoring in seismology, introduce further impediments to change [Ringler et al. 2015]. In summary, we see good though diverse support for initial creation in the upper hourglass with often sophisticated mapping to the lower hourglass, but subsequent workflow life cycle stages and reverse mappings remain a research goal.

4.3. Toward a Consistent Context for Data-Intensive Research

Every aspect of scientific workflow systems will be subject to improvement, and we note later some directions in which they will advance. However, we consider first responses to two pervasive pressures: (1) increasing data intensity partially driven by Kryder's law [Walter 2005] and new technologies (e.g., 3D Xpoint) and (2) increasing complexity from composing improvements and from workflow systems interworking.

The growing volumes of scientific data, the increased focus on data science, and the inexorable march of Kryder's law combine to overload the capacity of disk I/O—or more generally the bandwidth between RAM and external devices. This will drive increased adoption of data streaming between workflow stages, as these avoid a write out to disk followed by reading in, or double that traffic if files have to be moved. As long as stages can process a succession of data units and stream data units to subsequent stages, the code in the stages can remain resident, and the coupling can use in-RAM, local, or intersite communication mechanisms.[74] As with disk-mediated communication, moving data reduction as early as logically possible and employing lossless compression has benefits [Filgueira et al. 2014]. This approach mirrors shared-nothing and distributed query processing [Buil-Aranda et al. 2013] developed and refined in the database context [Stonebraker et al. 2013]. It is latent in the auto-iteration of Taverna and has been developed for Kepler [Kohler et al. 2012]. It is the model used by Meandre [Ács et al. 2010] and motivated the design of Dispel [Atkinson 2013]. It is the underpinning enactment model of dispel4py [Filguiera et al. 2016] and Bobolang [Falt et al. 2014]. As well as improving performance, this approach offers two extra benefits: (1) those working with live observations of time-dependent behavior—for example, observing the dynamics of natural phenomena or monitoring engineering or human systems, using data streaming workflows with the live data, and using exactly the same workflows with archived observations; and (2) as the interstage connection cost has been made small, it is feasible to include very simple stages—"fine-grained composition." Of course, large subsystems and services are also used in scientific methods encoded this way; for example, Python objects in dispel4py can wrap and interface with legacy code.

Workflow systems grow in complexity as extensions are added to accommodate more target DCIs, to handle more aspects of optimization, to automate frequently occurring actions, and to provide appropriate work environments for all three categories of experts contributing to data-intensive methods [Atkinson and Parsons 2013]. This growing complexity slows the rate at which these systems can respond to scientists' needs and exploit new opportunities. A two-pronged strategy is needed: (1) partitioning the system into manageable parts and (2) developing a formal framework to ensure parts and enhancements are consistent and work well together. The systems reported earlier have various partitioning strategies. Kepler uses an actor framework to partition parts and has a separate set of directors for orchestration. Pegasus uses three major partitions: the Wings framework for high-level composition, the planner and target-independent optimizers, and then the dynamic optimization during enactment delivered by DAGman and HTCondor. The WS-PGRADE/gUSE system has the partitions Data-avenue [Hajnal et al. 2014] and DCI-Bridge [Kozlovszky et al. 2014] to handle interfacing with data storage systems and with DCI, respectively. Independent repository management provides another partition, for example myExperiment and Wf4Ever (see earlier), or the WS-PGRADE/gUSE repository [Terstyánszky et al. 2014]. As workflow providers deploy more optimizations, as the paths between lightweight development and production environments are made smooth, and as interworking between workflow systems becomes more prevalent, these partitions will need to be kept small, with tightly defined consistent APIs. Agreeing on such an architectural structure is a research priority.

[74]Of course, capture of intermediary streams is needed for diagnostics during experiments and development.

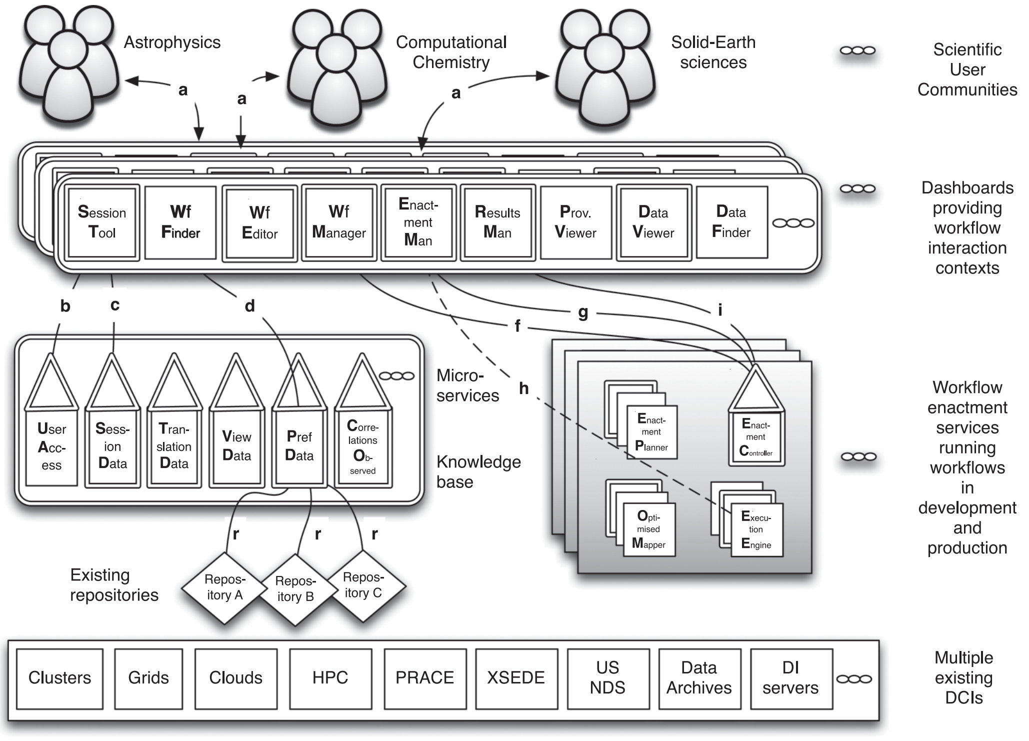

Fig. 10. Future partitioned and coupled workflow systems.

An envisaged future environment of services meeting the needs of scientists is shown in Figure 10. There are four major groups of subsystems: (1) the mobile device and web-enabled user interaction will deploy the latest web-enabled GUIs as a *Dashboard* and host a wide range of tools that can be tailored to the requirements of communities, groups, and individuals—this will handle interaction locally and depend on a wide range of underlying microservices; (2) the *knowledge base* accumulates workflows; workflow components; information about sessions, users, enactments, and community relationships; provenance records; and derivatives of these collected data, such as fragments that are often reused [Garijo 2015]—it supports interaction, including relaunching sessions; it supports security and controls; it provides information for optimizers and provides a basis for recommendations; it will integrate data from multiple application domains, communities, and technologies, interfacing with existing repositories and credential services; (3) the *enactment service* supports immediate execution for development and sophisticated choice of DCI targets with optimized mappings and dynamic optimization during execution for production; and (4) the *diverse DCI resources* are provided via many organizations, academic and commercial, and are shaped by many other requirements beyond scientific research and workflow enactment. There are many fronts on which scientific workflow systems will advance, and the capabilities they deliver to researchers will depend on combined advances.

5. CONCLUSIONS

Scientific communities have increasingly adopted workflow technologies to automate scientific methods. The emergence of data-driven science as the fourth paradigm has posed a new data challenge for scientific workflows, compounded by the increasing complexity and diversity of both applications and computing platforms.

In this article, we have proposed a taxonomy of WMSs that covers some aspects that have been overlooked in the related studies. Based on these architectural characterizations, we have reviewed seven prevalent WMSs that are widely adopted by research communities. These WMSs focused on different research communities with their specialist domains (e.g., life sciences, geosciences, high-energy physics, astronomy, and humanities) and slowly emerged into cross-disciplinary workflow infrastructures over the years. We took a bottom-up approach to analyzing these WMSs and presented a detailed architectural comparison that exposes their commonalities and diversity.

Section 4 identified a growing need for workflow systems that can be used directly by scientists to automate and formalize research methods. This will increase the pressure on workflow system research to make substantial advances in usability, interoperation, performance, and stability. It is crucial to reduce barriers associated with different modes of use and different technologies; this includes better access to distributed computing infrastructures, particularly those adapted to data-intensive requirements. Facilitating scientists experimenting with their own workflows will empower them to innovate and to exploit the growing data wealth to the fullest. The need for experts improving automated planning, mapping, and optimization and applying their own hand-crafted tuning in the cases where it is needed will continue. That will require improving the experts' productivity by equipping them with better tools and by automatically reusing their improvements.

There are deep technical challenges. A central one is the long-term growth of data storage and data acquisition outgrowing Moore's law by substantial factors. The interworking of WMSs will need to incorporate the advances in scientific database technologies and data-intensive middleware platforms. The extended data-intensive WMSs will need to be made accessible and comprehendible from web-enabled dynamic workplaces. This requires substantial advances in the conceptual and formal models describing the whole data-intensive infrastructure and the workflow languages that exploit it.

The emergent architecture for WMSs should support coalitions of the user communities associated with different WMSs as today's wealth of data and pressing societal challenges will require pooling intelligence and combining the best ideas and methods from many disciplines. The breadth of viewpoints and range of data and method ownership models will grow. For example, consider the food shortage challenge. Agricultural researchers need to combine all of the 'omic data from genomics to proteomics for every crop, animal, pest, and pathogen with data at many scales concerning soil type, aspect, climate and climate change, anthropogenic effects, farm management, and agro-pharma developments and create models to predict environmental effects, vulnerabilities, and yields at scales from individual plots to global [Rawlings 2014].

The power of data science drawing on today's growing wealth of data will only be realized if the WMS research rises to the challenge. Some of the issues to be faced have emerged in this review. A campaign is called for that builds both the theory and practice; that draws on all the intellectual and engineering powers of the many workflow experts, both in industry and academia; and that yields new families of mutually supporting, flexible, scalable, multiapplication, multicommunity, multipurpose, and sustainable workflow management systems with greatly increased power and platform independence and dramatically improved usability. They will need to be well integrated with data curation and all aspects of data sharing [Sansone et al. 2012]. We need a "moon-shot" culture where every effort and skill of the wider research community that will be needed to reach this goal is focused on achieving it; the problems encountered will need ingenious collaboration across discipline, organizational, technical, theoretical, and architectural boundaries.

ACKNOWLEDGMENTS

We thank James Cheney, University of Edinburgh (UoE), and Alessandro Spinuso of KNMI for alerting us to work on provenance and its standardization; Murray Cole, UoE, for encouraging our architectural comparison of WMSs; and Rosa Filgueira, UoE, and Oscar Corcho, Universidad Politécnica de Madrid, for their suggestions. We thank our reviewers for giving valuable insights. We particularly thank Paul Watson of Newcastle University for suggesting this article. Our work is supported by the Ministry of Education Malaysia (UMRG RP001F-13ICT and CG009-2013), the e-Science Core Programme Senior Research Fellow programme (UK EPSRC EP/D079829/1), and the EU projects ADMIRE, VERCE, and ENVRIplus (FP7 ICT 215024, FP7 RI 283543, and H2020 654182).

REFERENCES

B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, and others. 2016. Observation of gravitational waves from a binary Black Hole merger. *Phys. Rev. Lett.* 116, 6 (Feb. 2016), 061102. DOI:http://dx.doi.org/10.1103/PhysRevLett.116.061102

Mohamed Abouelhoda, Shadi Issa, and Moustafa Ghanem. 2012. Tavaxy: Integrating Taverna and Galaxy workflows with cloud computing support. *BMC Bioinformatics* 13, 1 (2012), 77. DOI:http://dx.doi.org/10.1186/1471-2105-13-77

David Abramson, Colin Enticott, and Ilkay Altinas. 2008. Nimrod/K: Towards massively parallel dynamic grid workflows. In *Proc. ACM/IEEE Conference on Supercomputing (SC'08)*. IEEE Press, Piscataway, NJ, USA, Article 24, 11 pages. DOI:http://dx.doi.org/10.1109/SC.2008.5215726

Bernie Ács, Xavier Llorà, Loretta Auvil, Boris Capitanu, David Tcheng, Mike Haberman, Limin Dong, Tim Wentling, and Michael Welge. 2010. A general approach to data-intensive computing using the Meandre component-based framework. In *Proc. 1st International Workshop on Workflow Approaches to New Data-centric Science (WANDS'10)*. ACM, Article 8, 12 pages. DOI:http://dx.doi.org/10.1145/1833398.1833406

Aashish N. Adhikari, Jian Peng, Michael Wilde, Jinbo Xu, Karl F. Freed, and Tobin R. Sosnick. 2012. Modeling large regions in proteins: Applications to loops, termini, and folding. *Protein Science* 21, 1 (Jan. 2012), 107–121. DOI:http://dx.doi.org/10.1002/pro.767

Chris Allan, Jean-Marie Burel, Josh Moore, Colin Blackburn, Melissa Linkert, Scott Loynton, Donald MacDonald, William J Moore, Carlos Neves, and others. 2012. OMERO: Flexible, model-driven data management for experimental biology. *Nature Methods* 9, 3 (March 2012), 245–253. DOI:http://dx.doi.org/10.1038/nmeth.1896

Ilkay Altintas, Oscar Barney, and Efrat Jaeger-Frank. 2006. Provenance collection support in the Kepler scientific workflow system. In *Provenance and Annotation of Data*. LNCS, Vol. 4145. 118–132. DOI:http://dx.doi.org/10.1007/11890850_14

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. 2010. A view of cloud computing. *Commun. ACM* 53, 4 (April 2010), 50–58. DOI:http://dx.doi.org/10.1145/1721654.1721672

Malcolm P. Atkinson. 2013. Data-Intensive thinking with Dispel. In *The Data Bonanza – Improving Knowledge Discovery for Science, Engineering and Business*, Malcolm P. Atkinson, Rob Baxter, Paolo Besana, Michelle Galea, Mark Parsons, Peter Brezany, Oscar Corcho, Jano van Hemert, and David Snelling (Eds.). John Wiley & Sons, Inc., Hoboken, NJ, USA, Chapter 4, 61–122. DOI:http://dx.doi.org/10.1002/9781118540343.ch4

Malcolm P. Atkinson, Michele Carpené, Emanuele Casarotti, Steffen Claus, Rosa Filgueira, Anton Frank, Michelle Galea, Tom Garth, André Gemünd, and others. 2015. VERCE delivers a productive e-Science environment for seismology research. In *Proc. IEEE International Conference on e-Science (e-Science 2015)*. DOI:http://dx.doi.org/10.1109/eScience.2015.38

Malcolm P. Atkinson and Mark Parsons. 2013. The Digital-Data Challenge. In *The Data Bonanza – Improving Knowledge Discovery for Science, Engineering and Business*, Malcolm P. Atkinson, Rob Baxter, Paolo Besana, Michelle Galea, Mark Parsons, Peter Brezany, Oscar Corcho, Jano van Hemert, and David Snelling (Eds.). John Wiley & Sons, Inc., Hoboken, NJ, USA, Chapter 1, 5–13. DOI:http://dx.doi.org/10.1002/9781118540343.ch1

Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. 2002. Models and issues in data stream systems. In *Proc. 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'02)*. ACM, New York, NY, USA, 1–16. DOI:http://dx.doi.org/10.1145/543613.543615

Roger Barga, Jared Jackson, Nelson Araujo, Dean Guo, Nitin Gautam, and Yogesh Simmhan. 2008. The trident scientific workflow workbench. In *Proc. e-Science'08*. IEEE Computer Society, Los Alamitos, CA, USA, 317–318. DOI:http://dx.doi.org/10.1109/eScience.2008.126

Adam Barker, Christopher D. Walton, and David Robertson. 2009. Choreographing web services. *IEEE Trans. on Services Computing* 2, 2 (April-June 2009), 152–166. DOI:http://dx.doi.org/10.1109/TSC.2009.8

Adam Barker, Jon B. Weissman, and Jano van Hemert. 2008. Orchestrating data-centric workflows. In *Proc. 8th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2008)*. IEEE Computer Society, 210–217. DOI:http://dx.doi.org/10.1109/CCGRID.2008.50

Jörg Becker, Michael zur Muehlen, and Marc Gille. 2002. Workflow application architectures: Classification and characteristics of workflow-based information systems. In *Workflow Handbook 2002*, Layna Fischer (Ed.). Future Strategies, 39–50.

Stephan Beisken, Thorsten Meinl, Bernd Wiswedel, Luis de Figueiredo, Michael Berthold, and Christoph Steinbeck. 2013. KNIME-CDK: Workflow-driven cheminformatics. *BMC Bioinformatics* 14, 1 (2013), 257. DOI:http://dx.doi.org/10.1186/1471-2105-14-257

Khalid Belhajjame, Jun Zhao, Daniel Garijo, Kristina Hettne, Raul Palma, Óscar Corcho, José-Manuel Gómez-Pérez, Sean Bechhofer, Graham Klyne, and Carole Goble. 2015. Using a suite of ontologies for preserving workflow-centric research objects. *Web Semantics: Science, Services and Agents on the World Wide Web* 32 (2015), 16–42. DOI:http://dx.doi.org/10.1016/j.websem.2015.01.003

G. Bruce Berriman, Ewa Deelman, Paul T. Groth, and Gideon Juve. 2010. The application of cloud computing to the creation of image mosaics and management of their provenance. In *Software and Cyberinfrastructure for Astronomy*, Nicole M. Radziwill and Alan Bridger (Eds.), Vol. 7740. SPIE, 77401F. DOI:http://dx.doi.org/10.1117/12.856486

Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. 2009. KNIME - The Konstanz information miner. *SIGKDD Explorations* 11, 1 (Nov. 2009), 26–31. DOI:http://dx.doi.org/10.1145/1656274.1656280

Shishir Bharathi, Ann Chervenak, Ewa Deelman, Gaurang Mehta, Mei-Hui Su, and Karan Vahi. 2008. Characterization of scientific workflows. In *Proc. Workflows for Science (WORKS'08)*. IEEE Computer Society, 1–10. DOI:http://dx.doi.org/10.1109/WORKS.2008.4723958

Daniel Blankenberg, Gregory Von Kuster, Nathaniel Coraor, Guruprasad Ananda, Ross Lazarus, Mary Mangan, Anton Nekrutenko, and James Taylor. 2010. *Galaxy: A Web-Based Genome Analysis Tool for Experimentalists*. John Wiley & Sons, Inc. DOI:http://dx.doi.org/10.1002/0471142727.mb1910s89

Peter A. Boncz, Martin L. Kersten, and Stefan Manegold. 2008. Breaking the memory wall in MonetDB. *Commun. ACM* 51, 12 (Dec. 2008), 77–85. DOI:http://dx.doi.org/10.1145/1409360.1409380

Shawn Bowers and Bertram Ludäscher. 2005. Actor-oriented design of scientific workflows. In *Conceptual Modeling – ER 2005*. LNCS, Vol. 3716. 369–384. DOI:http://dx.doi.org/10.1007/11568322_24

Shawn Bowers, Timothy McPhillips, Martin Wu, and Bertram Ludäscher. 2007. Project histories: Managing data provenance across collection-oriented scientific workflow runs. In *Data Integration in the Life Sciences*. LNCS, Vol. 4544. 122–138. DOI:http://dx.doi.org/10.1007/978-3-540-73255-6_12

P. Chris Broekema, Rob V. van Nieuwpoort, and Henri E. Bal. 2012. ExaScale high performance computing in the square kilometer array. In *Proc. Astro-HPC'12*. ACM, New York, NY, USA, 9–16. DOI:http://dx.doi.org/10.1145/2286976.2286982

Christopher Brooks, Edward A. Lee, Xiaojun Liu, Stephen Neuendorffer, Yang Zhao, and Haiyang Zheng. 2007. *Heterogeneous Concurrent Modeling and Design in Java (Volume 1: Introduction to Ptolemy II)*. Technical Report UCB/EECS-2007-7. EECS Department, University of California, Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-7.html.

Erik Brynjolfsson, Paul Hofmann, and John Jordan. 2010. Cloud computing and electricity: Beyond the utility model. *Commun. ACM* 53, 5 (May 2010), 32–34. DOI:http://dx.doi.org/10.1145/1735223.1735234

Tamás Budavári, László Dobos, and Alexander S. Szalay. 2013. SkyQuery: Federating astronomy archives. *Computing in Science & Engineering* 15, 3 (2013), 12–20. DOI:http://dx.doi.org/10.1109/MCSE.2013.41

Carlos Buil-Aranda, Marcelo Arenas, Oscar Corcho, and Axel Polleres. 2013. Federating queries in {SPARQL} 1.1: Syntax, semantics and evaluation. *Web Semantics: Science, Services and Agents on the World Wide Web* 18, 1 (2013), 1–17. DOI:http://dx.doi.org/10.1016/j.websem.2012.10.001 Special Section on the Semantic and Social Web.

Jacek Cała, Eyad Marei, Yaobo Xu, Kenji Takeda, and Paolo Missier. 2016. Scalable and efficient whole-exome data processing using workflows on the cloud. *Future Gener. Comput. Syst.* 65 (2016), 153–168. DOI:http://dx.doi.org/10.1016/j.future.2016.01.001

Scott Callaghan, Ewa Deelman, Dan Gunter, Gideon Juve, Philip Maechling, Christopher Brooks, Karan Vahi, Kevin Milner, Robert Graves, Edward Field, David Okaya, and Thomas Jordan. 2010. Scaling up workflow-based applications. *J. Comput. System Sci.* 76, 6 (2010), 428–446. DOI:http://dx.doi.org/10.1016/j.jcss.2009.11.005

Steven P. Callahan, Juliana Freire, Emanuele Santos, Carlos E. Scheidegger, Cláudio T. Silva, and Huy T. Vo. 2006. Managing the evolution of dataflows with VisTrails. In *Proc. 22nd International*

Conference on Data Engineering Workshops (ICDEW'06). IEEE Computer Society, Washington, DC, USA, 71. DOI:http://dx.doi.org/10.1109/ICDEW.2006.75

Sashi Kiran Challa, Marlon Pierce, and Suresh Marru. 2010. Integrating chemistry scholarship with web architectures, grid computing and semantic web. In *Proc. Gateway Computing Environments Workshop (GCE'10)*. 1–8. DOI:http://dx.doi.org/10.1109/GCE.2010.5676123

Matthew Chalmers. 2014. Large Hadron Collider: The big reboot. *Nature* 514 (2014), 158–160.

Jinjun Chen and Yun Yang. 2008. A taxonomy of grid workflow verification and validation. *Concurrency and Computation: Practice and Experience* 20, 4 (March 2008), 347–360. DOI:http://dx.doi.org/10.1002/cpe.1220

Weiwei Chen, Rafael Ferreira da Silva, Ewa Deelman, and Rizos Sakellariou. 2015. Using imbalance metrics to optimize task clustering in scientific workflow executions. *Future Gener. Comput. Syst.* 46 (2015), 69–84. DOI:http://dx.doi.org/10.1016/j.future.2014.09.014

Weiwei Chen and Ewa Deelman. 2011. Workflow overhead analysis and optimizations. In *Proc. WORKS'11*. ACM, New York, NY, USA, 11–20. DOI:http://dx.doi.org/10.1145/2110497.2110500

Daniel Crawl and Ilkay Altintas. 2008. A provenance-based fault tolerance mechanism for scientific workflows. In *Provenance and Annotation of Data and Processes*. LNCS, Vol. 5272. 152–159. 10.1007/978-3-540-89965-5_17

Víctor Cuevas-Vicenttín, Saumen Dey, Sven Köhler, Sean Riddle, and Bertram Ludäscher. 2012. Scientific workflows and provenance: Introduction and research opportunities. *Datenbank-Spektrum* 12, 3 (2012), 193–203. DOI:http://dx.doi.org/10.1007/s13222-012-0100-z

Sérgio Manuel Serra da Cruz, Maria Luiza M. Campos, and Marta Mattoso. 2009. Towards a taxonomy of provenance in scientific workflow management systems. In *Proc. 2009 IEEE Congress on Services - Part I (SERVICES'09)*. IEEE Computer Society, 259–266. DOI:http://dx.doi.org/10.1109/SERVICES-I.2009.18

David De Roure, Carole Goble, Sergejs Aleksejevs, Sean Bechhofer, Jiten Bhagat, Don Cruickshank, Paul Fisher, Nandkumar Kollara, Danius Michaelides, and others. 2010. The evolution of myExperiment. In *Proc. e-Science'10*. IEEE, 153–160. DOI:http://dx.doi.org/10.1109/eScience.2010.59

David De Roure, Carole Goble, and Robert Stevens. 2009. The design and realisation of the myExperiment virtual research environment for social sharing of workflows. *Future Gener. Comput. Syst.* 25, 5 (2009), 561–567. DOI:http://dx.doi.org/10.1016/j.future.2008.06.010

David De Roure, Kevin R. Page, Benjamin Fields, Tim Crawford, J. Stephen Downie, and Ichiro Fujinaga. 2011. An e-research approach to web-scale music analysis. *Phil. Trans. R. Soc. A* 369, 1949 (Aug. 2011), 3300–3317. DOI:http://dx.doi.org/10.1098/rsta.2011.0171

Ewa Deelman. 2010. Grids and clouds: Making workflow applications work in heterogeneous distributed environments. *International Journal of High Performance Computing Applications* 24, 3 (Aug. 2010), 284–298. DOI:http://dx.doi.org/10.1177/1094342009356432

Ewa Deelman, Scott Callaghan, Edward Field, Hunter Francoeur, Robert Graves, Nitin Gupta, Vipin Gupta, Thomas H. Jordan, Carl Kesselman, and others. 2006. Managing large-scale workflow execution from resource provisioning to provenance tracking: The cybershake example. In *Proc. e-Science'06*. 14. DOI:http://dx.doi.org/10.1109/E-SCIENCE.2006.261098

Ewa Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. 2009. Workflows and e-Science: An overview of workflow system features and capabilities. *Future Gener. Comput. Syst.* 25, 5 (May 2009), 528–540. DOI:http://dx.doi.org/10.1016/j.future.2008.06.012

Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip J. Maechling, Rajiv Mayani, Weiwei Chen, Rafael Ferreira da Silva, Miron Livny, and Kent Wenger. 2015. Pegasus, a workflow management system for science automation. *Future Gener. Comput. Syst.* 46 (2015), 17–35. DOI:http://dx.doi.org/10.1016/j.future.2014.10.008

Ewa Deelman, Karan Vahi, Mats Rynge, Gideon Juve, Rajiv Mayani, and Rafael Ferreira da Silva. 2016. Pegasus in the cloud: Science automation through workflow technologies. *IEEE Internet Computing* 20, 1 (Jan. 2016), 70–76. DOI:http://dx.doi.org/10.1109/MIC.2016.15

László Dobos, István Csabai, Alexander S. Szalay, Tamás Budavári, and Nolan Li. 2013. Graywulf: A platform for federated scientific databases and services. In *Proc. 25th International Conference on Scientific and Statistical Database Management (SSDBM)*. ACM, New York, NY, USA, Article 30, 12 pages. DOI:http://dx.doi.org/10.1145/2484838.2484863

Rion Dooley, Kent Milfeld, Chona Guiang, Sudhakar Pamidighantam, and Gabrielle Allen. 2006. From proposal to production: Lessons learned developing the computational chemistry grid cyberinfrastructure. *Journal of Grid Computing* 4, 2 (2006), 195–208. DOI:http://dx.doi.org/10.1007/s10723-006-9043-7

Lei Dou, Daniel Zinn, Timothy McPhillips, Sven Kohler, Sean Riddle, Shawn Bowers, and Bertram Ludäscher. 2011. Scientific workflow design 2.0: Demonstrating streaming data collections in Kepler. In *Proc. IEEE ICDE'11*. 1296–1299. DOI:http://dx.doi.org/10.1109/ICDE.2011.5767938

Johan Eker, Jörn W. Janneck, Edward A. Lee, Jie Liu, Xiaojun Liu, Jozsef Ludvig, Stephen Neuendorffer, Sonia Sachs, and Yuhong Xiong. 2003. Taming heterogeneity - the Ptolemy approach. *Proc. IEEE* 91, 1 (Jan. 2003), 127–144. DOI:http://dx.doi.org/10.1109/JPROC.2002.805829

Erik Elmroth, Francisco Hernández, and Johan Tordsson. 2010. Three fundamental dimensions of scientific workflow interoperability: Model of computation, language, and execution environment. *Future Gener. Comput. Syst.* 26, 2 (Feb. 2010), 245–256. DOI:http://dx.doi.org/10.1016/j.future.2009.08.011

Wolfgang Emmerich, Ben Butchart, Liang Chen, Bruno Wassermann, and Sarah Price. 2005. Grid service orchestration using the business process execution language (BPEL). *Journal of Grid Computing* 3, 3 (Sept. 2005), 283–304. DOI:http://dx.doi.org/10.1007/s10723-005-9015-3

EU Parliament. 2007. Directive 2007/2/EC of the European parliament and of the council of 14 march 2007 establishing an infrastructure for spatial information in the european community (INSPIRE). *Official Journal of the European Union* 50, L108 (April 2007).

Thomas Fahringer, Radu Prodan, Rubing Duan, Jüurgen Hofer, Farrukh Nadeem, Francesco Nerieri, Stefan Podlipnig, Jun Qin, Mumtaz Siddiqui, and others. 2007. ASKALON: A development and grid computing environment for scientific workflows. In *Workflows for e-Science: Scientific Workflows for Grids*, Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields (Eds.). Springer London, 450–471. DOI:http://dx.doi.org/10.1007/978-1-84628-757-2_27

Zbyněk Falt, David Bednárek, Martin Kruliš, Jakub Yaghob, and Filip Zavoral. 2014. Bobolang: A language for parallel streaming applications. In *Proc. HPDC'14*. ACM, New York, NY, USA, 311–314. DOI:http://dx.doi.org/10.1145/2600212.2600711

Rosa Filgueira, Malcolm Atkinson, Yusuke Tanimura, and Isao Kojima. 2014. Applying selectively parallel I/O compression to parallel storage systems. In *Euro-Par 2014 Parallel Processing*. LNCS, Vol. 8632. 282–293. DOI:http://dx.doi.org/10.1007/978-3-319-09873-9_24

Rosa Filguiera, Amrey Krause, Malcolm Atkinson, Iraklis Klampanos, and Alexander Moreno. 2016. dispel4py: A python framework for data-intensive scientific computing. *International Journal of High Performance Computing Applications* (2016), 1–19. DOI:http://dx.doi.org/10.1177/1094342016649766

Ian Foster, Jens Vöckler, Michael Wilde, and Yong Zhao. 2002. Chimera: A virtual data system for representing, querying, and automating data derivation. In *Proc. SSDBM'02*. 37–46. DOI:http://dx.doi.org/10.1109/SSDM.2002.1029704

Scott W. French and Barbara Romanowicz. 2015. Broad plumes rooted at the base of the Earth's mantle beneath major hotspots. *Nature* 525, 7567 (03 09 2015), 95–99. 10.1038/nature14876.

Dennis Gannon. 2007. Component architectures and services: From application construction to scientific workflows. In *Workflows for e-Science: Scientific Workflows for Grids*, Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields (Eds.). Springer London, 174–189. DOI:http://dx.doi.org/10.1007/978-1-84628-757-2_12

Daniel Garijo. 2015. *Mining Abstractions in Scientific Workflows*. Ph.D. Dissertation. Departamento de Inteligencia Artficial Escuela Técnica Superior de Ingenieros Informáticos, Madrid, Spain.

Daniel Garijo, Facultad De Informática, and Yolanda Gil. 2012. Towards Open Publication of Reusable Scientific Workflows: Abstractions, Standards and Linked Data. Technical Report. (Jan. 2012).

Sandra Gesing, Malcolm Atkinson, Rosa Filgueira, Ian Taylor, Andrew Jones, Vlado Stankovski, Chee Sun Liew, Alessandro Spinuso, Gabor Terstyanszky, and Peter Kacsuk. 2014. Workflows in a dashboard: A new generation of usability. In *Proc. WORKS'14*. IEEE Press, Piscataway, NJ, USA, 82–93. DOI:http://dx.doi.org/10.1109/WORKS.2014.6

Jayeeta Ghosh, Suresh Marru, Nikhil Singh, Kenno Vanomesslaeghe, Ye Fan, and Sudhakar Pamidighantam. 2011. Molecular parameter optimization gateway (ParamChem): Workflow management through teragrid ASTA. In *Proc. TeraGrid (TG'11)*. ACM, 35:1–35:8. DOI:http://dx.doi.org/10.1145/2016741.2016779

Yolanda Gil, Jihie Kim, Varun Ratnakar, and Ewa Deelman. 2006. Wings for Pegasus: A semantic approach to creating very large scientific workflows. In *Proc. Workshop on OWL: Experiences and Directions (OWLED'06)*, Vol. 216.

Edward Givelberg, Alexander Szalay, Kalin Kanov, and Randal Burns. 2011. An architecture for a data-intensive computer. In *Proc. Network Aware Data Management (NDM'11)*. ACM, New York, NY, USA, 57–64. DOI:http://dx.doi.org/10.1145/2110217.2110226

Carole Goble and David De Roure. 2009. The impact of workflow tools on data-centric research. In *The Fourth Paradigm: Data-Intensive Scientific Discovery*, Tony Hey, Stewart Tansley, and Kristin Tolle (Eds.). Microsoft, 137–145.

Katharina Görlach, Mirko Sonntag, Dimka Karastoyanova, Frank Leymann, and Michael Reiter. 2011. Conventional workflow technology for scientific simulation. In *Guide to e-Science*. 323–352. DOI:http://dx.doi.org/10.1007/978-0-85729-439-5_12

Ian Gorton, Paul Greenfield, Alex Szalay, and Roy Williams. 2008. Data-intensive computing in the 21st century. *Computer* 41, 4 (April 2008), 30–32. DOI:http://dx.doi.org/10.1109/MC.2008.122

Jim Gray. 2009. Jim gray on escience: A transformed scientific method. In *The Fourth Paradigm: Data-Intensive Scientific Discovery*, Tony Hey, Stewart Tansley, and Kristin Tolle (Eds.). Microsoft, xix–xxxiii.

Paul Grefen and Jochem Vonk. 2006. A taxonomy of transactional workflow support. *International Journal of Cooperative Information Systems* 15, 1 (March 2006), 87–118. DOI:http://dx.doi.org/10.1142/S021884300600130X

Paul Groth, Yolanda Gil, James Cheney, and Simon Miles. 2012. Requirements for provenance on the web. *International Journal of Digital Curation* 7, 1 (2012), 39–55.

Yunhong Gu and Robert L. Grossman. 2009. Sector and sphere: The design and implementation of a high-performance data cloud. *Phil. Trans. R. Soc. A* 367, 1897 (June 2009), 2429–2445. DOI:http://dx.doi.org/10.1098/rsta.2009.0053

Thilina Gunarathne, Chathura Herath, Eran Chinthaka, and Suresh Marru. 2009. Experience with adapting a WS-BPEL runtime for escience workflows. In *Proc. GCE'09*. ACM, 7:1–7:10. DOI:http://dx.doi.org/10.1145/1658260.1658270

Ákos Hajnal, Zoltán Farkas, Péter Kacsuk, and Tamás Pintér. 2014. Remote storage resource management in WS-PGRADE/gUSE. In *Science Gateways for Distributed Computing Infrastructures: Development Framework and Exploitation by Scientific User Communities*, Péter Kacsuk (Ed.). Springer, Chapter 5, 69–81. DOI:http://dx.doi.org/10.1007/978-3-319-11268-8_5

Mihael Hategan, Justin Wozniak, and Ketan Maheshwari. 2011. Coasters: Uniform resource provisioning and access for clouds and grids. In *Proc. 4th IEEE International Conference on Utility and Cloud Computing (UCC'11)*. IEEE Computer Society, 114–121. DOI:http://dx.doi.org/10.1109/UCC.2011.25

George Heald, Michael Bell, Andreas Horneffer, André Offringa, Roberto Pizzo, Sebastiaan van der Tol, Reinout van Weeren, Joris van Zwieten, James Anderson, and others. 2011. LOFAR: Recent imaging results and future prospects. *Journal of Astrophysics and Astronomy* 32, 4 (Dec. 2011), 1–10. DOI:http://dx.doi.org/10.1007/s12036-011-9125-1

Tom Heath and Christian Bizer. 2011. *Linked Data: Evolving the Web into a Global Data Space* (1st ed.). Number 1-136 in Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool.

Tony Hey, Stewart Tansley, and Kristin Tolle (Eds.). 2009. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research.

Interagency Working Group on Digital Data. 2009. *Harnessing the Power of Digital Data for Science and Society: Report of the Interagency Working Group on Digital Data to the National Science and Technology Council*. Technical Report. Executive office of the President, Office of Science and Technology, USA.

Gideon Juve and Ewa Deelman. 2010. Scientific workflows and clouds. *Crossroads* 16, 3 (March 2010), 14–18. DOI:http://dx.doi.org/10.1145/1734160.1734166

Péter Kacsuk (Ed.). 2014. *Science Gateways for Distributed Computing Infrastructures: Development Framework and Exploitation by Scientific User Communities*. DOI:http://dx.doi.org/10.1007/978-3-319-11268-8

Peter Kacsuk, Zoltan Farkas, Miklos Kozlovszky, Gabor Hermann, Akos Balasko, Krisztian Karoczkai, and Istvan Marton. 2012. WS-PGRADE/gUSE Generic DCI gateway framework for a large variety of user communities. *Journal of Grid Computing* 10, 4 (2012), 601–630. DOI:http://dx.doi.org/10.1007/s10723-012-9240-5

Peter Kacsuk, Gabor Terstyánszky, Ákos Balaskó, , Krisztian Karóczkai, and Zoltan Farkas. 2014. Executing multi-workflow simulations on mixed cloud and grid infrastructure using the SHIWA and SCI-BUS technology. In *Cloud Computing and Big Data*, C. Catlett, W. Gentzsch, L. Grandinetti, and G. Joubert (Eds.). Ios Pr Inc, 141–162.

Douglas B. Kell and Stephen G. Oliver. 2004. Here is the evidence, now what is the hypothesis? The complementary roles of inductive and hypothesis-driven science in the post-genomic era. *BioEssays* 26, 1 (Jan. 2004), 99–105. DOI:http://dx.doi.org/10.1002/bies.10385

Steve Kelling, Daniel Fink, Wesley Hochachka, Ken Rosenberg, Robert Cook, Theodoros Damoulas, Claudio Silva, and William Michener. 2013. Estimating species distributions – across space, through time and with features of the environment. In *The Data Bonanza – Improving Knowledge Discovery for Science, Engineering and Business*, Malcolm P. Atkinson, Rob Baxter, Paolo Besana, Michelle Galea, Mark Parsons, Peter Brezany, Oscar Corcho, Jano van Hemert, and David Snelling (Eds.). John Wiley & Sons Inc., Hoboken, NJ, USA, Chapter 22, 441–458. DOI:http://dx.doi.org/10.1002/9781118540343.ch22

Jihie Kim, Ewa Deelman, Yolanda Gil, Gaurang Mehta, and Varun Ratnakar. 2008. Provenance trails in the Wings/Pegasus system. *Concurrency and Computation: Practice and Experience* 20, 5 (April 2008), 587–597. DOI:http://dx.doi.org/10.1002/cpe.1228

Hoyt Koepke. 2014. *Why Python Rocks for Research*. Technical Report. University of Washington.

Sven Kohler, Supriya Gulati, Gongjing Cao, Quinn Hart, and Bertram Ludascher. 2012. Sliding window calculations on streaming data using the Kepler scientific workflow system. *Procedia Computer Science* 9, 0 (2012), 1639–1646. DOI:http://dx.doi.org/10.1016/j.procs.2012.04.181

Vladimir Korkhov, Dagmar Krefting, Tamas Kukla, Gabor Z. Terstyánszky, Matthan W. A. Caan, and Silvia D. Olabarriaga. 2013. Exploring workflow interoperability for neuroimage analysis on the SHIWA platform. *Journal of Grid Computing* 11, 3 (2013), 505–522. DOI:http://dx.doi.org/10.1007/s10723-013-9262-7

Miklos Kozlovszky, Krisztián Karóczkai, István Márton, Péter Kacsuk, and Tibor Gottdank. 2014. DCI Bridge: Executing WS-PGRADE workflows in distributed computing infrastructures. In *Science Gateways for Distributed Computing Infrastructures: Development Framework and Exploitation by Scientific User Communities*, Péter Kacsuk (Ed.). Springer, Chapter 4, 51–67. DOI:http://dx.doi.org/10.1007/978-3-319-11268-8_4

Michael Litzkow, Miron Livny, and Matthew Mutka. 1988. Condor - A hunter of idle workstations. In *Proc. 8th International Conference of Distributed Computing Systems*. IEEE Computer Society Press, 104–111. DOI:http://dx.doi.org/10.1109/DCS.1988.12507

Xavier Llorà, Bernie Ács, Loretta S. Auvil, Boris Capitanu, Michael E. Welge, and David E. Goldberg. 2008. Meandre: Semantic-driven data-intensive flows in the clouds. In *Proc. e-Science'08*. 238–245. DOI:http://dx.doi.org/10.1109/eScience.2008.172

Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. 2006. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience* 18, 10 (August 2006), 1039–1065. DOI:http://dx.doi.org/10.1002/cpe.994

Bertram Ludäscher, Mathias Weske, Timothy McPhillips, and Shawn Bowers. 2009. Scientific workflows: Business as usual? In *Business Process Management*. LNCS, Vol. 5701. 31–47. DOI:http://dx.doi.org/10.1007/978-3-642-03848-8_4

Philip Maechling, Ewa Deelman, Li Zhao, Robert Graves, Gaurang Mehta, Nitin Gupta, John Mehringer, Carl Kesselman, Scott Callaghan, David Okaya, Hunter Francoeur, Vipin Gupta, Yifeng Cui, Karan Vahi, Thomas Jordan, and Edward Field. 2007. SCEC cybershake workflows—Automating probabilistic seismic hazard analysis calculations. In *Workflows for e-Science: Scientific Workflows for Grids*, Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields (Eds.). Springer London, 143–163. DOI:http://dx.doi.org/10.1007/978-1-84628-757-2_10

Ketan Maheshwari, Alex Rodriguez, David Kelly, Ravi Madduri, Justin Wozniak, Michael Wilde, and Ian Foster. 2013. Enabling multi-task computation on Galaxy-based gateways using swift. In *Proc. IEEE International Conference on Cluster Computing (CLUSTER 2013)*. 1–3. DOI:http://dx.doi.org/10.1109/CLUSTER.2013.6702701

Suresh Marru, Lahiru Gunathilake, Chathura Herath, Patanachai Tangchaisin, Marlon Pierce, Chris Mattmann, Raminder Singh, Thilina Gunarathne, Eran Chinthaka, and others. 2011. Apache airavata: A framework for distributed applications and computational workflows. In *Proc. GCE'11*. ACM, 21–28. DOI:http://dx.doi.org/10.1145/2110486.2110490

Suresh Marru, Marlon Pierce, Sudhakar Pamidighantam, and Chathuri Wimalasena. 2015. Apache airavata as a laboratory: Architecture and case study for component-based gateway middleware. In *Proc. SCREAM'15*. 19–26. DOI:http://dx.doi.org/10.1145/2753524.2753529

Paul Martin and Gagarine Yaikhom. 2013. Definition of the DISPEL language. In *The Data Bonanza – Improving Knowledge Discovery for Science, Engineering and Business*, Malcolm P. Atkinson, Rob Baxter, Paolo Besana, Michelle Galea, Mark Parsons, Peter Brezany, Oscar Corcho, Jano van Hemert, and David Snelling (Eds.). John Wiley & Sons Inc., Hoboken, NJ, USA, Chapter 10, 203–236. DOI:http://dx.doi.org/10.1002/9781118540343.ch10

Cherian Mathew, Anton Güntsch, Matthias Obst, Saverio Vicario, Robert Haines, Alan Williams, Yde de Jong, and Carole Goble. 2014. A semi-automated workflow for biodiversity data retrieval, cleaning, and quality control. *Biodiversity Data Journal* 2 (Dec. 2014), e4221. DOI:http://dx.doi.org/10.3897/BDJ.2.e4221

Michael McLennan, Steven Clark, Ewa Deelman, Mats Rynge, Karan Vahi, Frank McKenna, Derrick Kearney, and Carol Song. 2015. HUBzero and Pegasus: Integrating scientific workflows into science gateways. *Concurrency and Computation: Practice and Experience* 27, 2 (2015), 328–343. DOI:http://dx.doi.org/10.1002/cpe.3257

Michael McLennan and Rick Kennell. 2010. HUBzero: A platform for dissemination and collaboration in computational science and engineering. *Computing in Science Engineering* 12, 2 (March 2010), 48–53. DOI:http://dx.doi.org/10.1109/MCSE.2010.41

Timothy M. McPhillips and Shawn Bowers. 2005. An approach for pipelining nested collections in scientific workflows. *SIGMOD Record* 34, 3 (Sept. 2005), 12–17. DOI:http://dx.doi.org/10.1145/1084805.1084809

William Michener, James Beach, Shawn Bowers, Laura Downey, Matthew Jones, Bertram Ludäscher, Deana Pennington, Arcot Rajasekar, Samantha Romanello, Mark Schildhauer, Dave Vieglais, and Jianting Zhang. 2005. Data integration and workflow solutions for ecology. In *Data Integration in the Life Sciences*. LNCS, Vol. 3615. 734–734. DOI:http://dx.doi.org/10.1007/11530084_32

Paolo Missier, Bertram Ludascher, Shawn Bowers, Saumen Dey, Anandarup Sarkar, Biva Shrestha, Ilkay Altintas, Manish Kumar Anand, and Carole Goble. 2010a. Linking multiple workflow provenance traces for interoperable collaborative science. In *WORKS'10*. 1–8. DOI:http://dx.doi.org/10.1109/WORKS.2010.5671861

Paolo Missier, Bertram Ludäscher, Saumen C. Dey, Michael Wang, Timothy M. McPhillips, Shawn Bowers, Michael Agun, and Ilkay Altintas. 2012. Golden trail: Retrieving the data history that matters from a comprehensive provenance repository. *IJDC* 7, 1 (2012), 139–150. DOI:http://dx.doi.org/10.2218/ijdc.v7i1.221

Paolo Missier, Stian Soiland-Reyes, Stuart Owen, Wei Tan, Alexandra Nenadic, Ian Dunlop, Alan Williams, Tom Oinn, and Carole Goble. 2010b. Taverna, Reloaded. In *Scientific and Statistical Database Management*. LNCS, Vol. 6187. 471–481. DOI:http://dx.doi.org/10.1007/978-3-642-13818-8_33

Fiona Murphy, Publishing Data Workflows WG, Theodora Bloom, Sunje Dallmeier-Tiessen, Claire C. Austin, Angus Whyte, Jonathan Tedds, Amy Nurnberger, Lisa Raymond, Martina Stockhause, and Mary Vardigan. 2015. WDS-RDA Publishing Data Workflows Working Group Analysis sheet. (June 2015). DOI:http://dx.doi.org/10.5281/zenodo.19107

James Myers, Margaret Hedstrom, Dharma Akmon, Sandy Payette, Beth A. Plale, Inna Kouper, Scott McCaulay, Robert McDonald, Isuru Suriarachchi, and others. 2015. Towards sustainable curation and preservation. In *Proc. e-Science'15*. 526–535. DOI:http://dx.doi.org/10.1109/eScience.2015.56

Michael L. Norman and Allan Snavely. 2010. Accelerating data-intensive science with Gordon and Dash. In *Proc. TG'10*. ACM, New York, NY, USA, Article 14, 7 pages. DOI:http://dx.doi.org/10.1145/1838574.1838588

Thomas Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew Pocock, Anil Wipat, and Peter Li. 2004. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 20, 17 (Nov. 2004), 3045–3054. DOI:http://dx.doi.org/10.1093/bioinformatics/bth361

Tom Oinn, Mark Greenwood, Matthew Addis, M. Nedim Alpdemir, Justin Ferris, Kevin Glover, Carole Goble, Antoon Goderis, Duncan Hull, and others. 2006. Taverna: Lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience* 18, 10 (2006), 1067–1100. DOI:http://dx.doi.org/10.1002/cpe.993

Tom Oinn, Peter Li, Douglas B. Kell, Carole Goble, Antoon Goderis, Mark Greenwood, Duncan Hull, Robert Stevens, Daniele Turi, and Jun Zhao. 2007. Taverna/myGrid: Aligning a workflow system with the life sciences community. In *Workflows for e-Science: Scientific Workflows for Grids*, Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields (Eds.). Springer London, 300–319. DOI:http://dx.doi.org/10.1007/978-1-84628-757-2_19

Ioan Raicu, Yong Zhao, Catalin Dumitrescu, Ian Foster, and Mike Wilde. 2007. Falkon: A fast and light-weight tasK executiON framework. In *Proc. SC'07*. ACM, New York, NY, USA, Article 43, 12 pages. DOI:http://dx.doi.org/10.1145/1362622.1362680

Christopher Rawlings. 2014. Big data in the agricultural and ecological sciences — a growing challenge. Keynote EGI Community Forum 2014. (May 2014).

A. T. Ringler, M. T. Hagerty, J. Holland, A. Gonzales, L. S. Gee, J. D. Edwards, D. Wilson, and A. M. Baker. 2015. The data quality analyzer: A quality control program for seismic data. *Computers & Geosciences* 76 (2015), 96–111.

David Rogers, Ian Harvey, Tram Truong Huu, Kieran Evans, Tristan Glatard, Ibrahim Kallel, Ian Taylor, Johan Montagnat, Andrew Jones, and Andrew Harrison. 2013. Bundle and pool architecture for multi-language, robust, scalable workflow executions. *Journal of Grid Computing* 11, 3 (2013), 457–480.

John W. Romein, Jan David Mol, Rob V. van Nieuwpoort, and P. Chris Broekema. 2011. Processing LOFAR telescope data in real time on a blue Gene/P supercomputer. In *General Assembly and Scientific Symposium, 2011 XXXth URSI*. 1–4. DOI:http://dx.doi.org/10.1109/URSIGASS.2011.6051270

Susanna-Assunta Sansone, Philippe Rocca-Serra, Dawn Field, Eamonn Maguire, Chris Taylor, Oliver Hofmann, Hong Fang, Steffen Neumann, Weida Tong, and others. 2012. Toward interoperable bioscience data. *Nat. Genet.* 44, 2 (02 2012), 121–126. DOI:http://dx.doi.org/10.1038/ng.1054

Idafen Santana-Perez, Rafael Ferreira da Silva, Mats Rynge, Ewa Deelman, María S. Pérez-Hernández, and Oscar Corcho. 2016. Reproducibility of execution environments in computational science using Semantics and Clouds. *Future Gener. Comput. Syst.* 67 (2016), 354–367. DOI:http://dx.doi.org/10.1016/j.future.2015.12.017

Matthew Shields. 2007. Control- versus data-driven workflows. In *Workflows for e-Science: Scientific Workflows for Grids*, Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields (Eds.). Springer London, 167–173. DOI:http://dx.doi.org/10.1007/978-1-84628-757-2_11

Yogesh L. Simmhan, Roger Barga, Catharine van Ingen, Ed Lazowska, and Alex Szalay. 2009. Building the trident scientific workflow workbench for data management in the cloud. In *Proc. 3rd International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP'09)*. 41–50. DOI:http://dx.doi.org/10.1109/ADVCOMP.2009.14

Aleksander Slominski. 2007. Adapting BPEL to scientific workflows. In *Workflows for e-Science: Scientific Workflows for Grids*, Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields (Eds.). Springer London, 208–226. DOI:http://dx.doi.org/10.1007/978-1-84628-757-2_14

Alessandro Spinuso, Rosa Fligueira, Malcolm Atkinson, and Andre Gemuend. 2016. Visualisation methods for large provenance collections in data-intensive collaborative platforms. In *Geophysical Research Abstracts - EGU General Assembly 2016*, Vol. 18.

Sudarshan Srinivasan, Gideon Juve, Rafael Ferreira da Silva, Karan Vahi, and Ewa Deelman. 2014. A cleanup algorithm for implementing storage constraints in scientific workflow executions. In *Proc. WORKS'14*. IEEE Press, 41–49. DOI:http://dx.doi.org/10.1109/WORKS.2014.8

Tiberiu Stef-Praun, Benjamin Clifford, Ian Foster, Uri Hasson, Mihael Hategan, Steven L. Small, Michael Wilde, and Yong Zhao. 2007. Accelerating medical research using the swift workflow system. *Studies in Health Technology and Informatics* 126 (2007), 207–216.

Michael Stonebraker, Jacek Becla, David J. DeWitt, Kian-Tat Lim, David Maier, Oliver Ratzesberger, and Stanley B. Zdonik. 2009. Requirements for science data bases and SciDB. In *Proc. Biennial Conference on Innovative Data Systems Research (CIDR'09)*.

Michael Stonebraker, Paul Brown, Donghui Zhang, and Jacek Becla. 2013. SciDB: A database management system for applications with complex analytics. *Computing in Science & Engineering* 15, 3 (2013), 54–62.

Ian Taylor, Matthew Shields, Ian Wang, and Andrew Harrison. 2007a. The Triana workflow environment: Architecture and applications. In *Workflows for e-Science: Scientific Workflows for Grids*, Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields (Eds.). Springer London, 320–339. DOI:http://dx.doi.org/10.1007/978-1-84628-757-2_20

Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields. 2007b. Workflows for e-Science: Scientific workflows for grids. Springer London. DOI:http://dx.doi.org/10.1007/978-1-84628-757-2

Gabor Terstyánszky, Edward Michniak, Tamás Kiss, and Ákos Balaskó. 2014. Sharing science gateway artefacts through repositories. In *Science Gateways for Distributed Computing Infrastructures: Development Framework and Exploitation by Scientific User Communities*. Springer, Chapter 9, 123–135. DOI:http://dx.doi.org/10.1007/978-3-319-11268-8_9

Douglas Thain, Todd Tannenbaum, and Miron Livny. 2005. Distributed computing in practice: The Condor experience. *Concurrency and Computation: Practice and Experience* 17, 2-4 (2005), 323–356. DOI:http://dx.doi.org/10.1002/cpe.938

Thomas D. Uram, Michael E. Papka, Mark Hereld, and Michael Wilde. 2011. A solution looking for lots of problems: generic portals for science infrastructure. In *Proc. TG'11*. ACM, New York, NY, USA, Article 44, 7 pages. DOI:http://dx.doi.org/10.1145/2016741.2016788

Wil M. P. van der Aalst and Arthur H. M. ter Hofstede. 2014. Workflow Patterns. http://www.workflowpatterns.com. (2014).

Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. 2003. Workflow Patterns. *Distributed and Parallel Databases* 14, 1 (July 2003), 5–51. DOI:http://dx.doi.org/10.1023/A:1022883727209

Jens Vöckler, Gaurang Mehta, Yong Zhao, Ewa Deelman, and Michael Wilde. 2006. Kickstarting Remote Applications. In *Second International Workshop on Grid Computing Environments*.

Gregor von Laszewski and Mike Hategan. 2005. Workflow Concepts of the Java CoG Kit. *Journal of Grid Computing* 3, 3 (Sept. 2005), 239–258. DOI:http://dx.doi.org/10.1007/s10723-005-9013-5

Chip Walter. 2005. Kryder's Law: The doubling of processor speed every 18 months is a snail's pace compared with rising hard-disk capacity, and Mark Kryder plans to squeeze in even more bits. *Scientific American* (August 2005), 32–33.

Hongbing Wang, Joshua Zhexue Huang, Yuzhong Qu, and Junyuan Xie. 2004. Web services: Problems and future directions. *Web Semantics: Science, Services and Agents on the World Wide Web* 1, 3 (April 2004), 309–320. DOI:http://dx.doi.org/10.1016/j.websem.2004.02.001

Marek Wieczorek, Andreas Hoheisel, and Radu Prodan. 2009. Towards a general model of the multi-criteria workflow scheduling on the grid. *Future Gener. Comput. Syst.* 25, 3 (March 2009), 237–256. DOI:http://dx.doi.org/10.1016/j.future.2008.09.002

Michael Wilde, Ian Foster, Kamil Iskra, Pete Beckman, Zhao Zhang, Allan Espinosa, Mihael Hategan, Ben Clifford, and Ioan Raicu. 2009. Parallel scripting for applications at the petascale and beyond. *Computer* 42, 11 (Nov. 2009), 50–60. DOI:http://dx.doi.org/10.1109/MC.2009.365

Matthew Woitaszek, John M. Dennis, and Taleena R. Sine. 2011. Parallel high-resolution climate data analysis using swift. In *Proc. ACM International Workshop on Many Task Computing on Grids and Supercomputers (MTAGS'11)*. ACM, New York, NY, USA, 5–14. DOI:http://dx.doi.org/10.1145/2132876.2132882

Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, and others. 2013. The Taverna workflow suite: Designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research* 41, W1 (2013), W557–W561. DOI:http://dx.doi.org/10.1093/nar/gkt328

Justin M. Wozniak, Timothy G. Armstrong, Ketan Maheshwari, Ewing L. Lusk, Daniel S. Katz, Michael Wilde, and Ian T. Foster. 2013a. Turbine: A distributed-memory dataflow engine for high performance many-task applications. *Fundamenta Informaticae* 128, 3 (01 2013), 337–366. DOI:http://dx.doi.org/10.3233/FI-2013-949

Justin M. Wozniak, Timothy G. Armstrong, Michael Wilde, Daniel S. Katz, Ewing Lusk, and Ian T. Foster. 2013b. Swift/T: Large-scale application composition via distributed-memory dataflow processing. In *Proc. IEEE/ACM CCGRID'13*. 95–102. DOI:http://dx.doi.org/10.1109/CCGrid.2013.99

Wenjun Wu, Thomas Uram, Michael Wilde, Mark Hereld, and Michael E. Papka. 2010. Accelerating science gateway development with Web 2.0 and Swift. In *Proc. TG'10*. ACM, New York, NY, USA, Article 23, 7 pages. DOI:http://dx.doi.org/10.1145/1838574.1838597

Youngik Yang, Jong Youl Choi, Chathura Herath, Suresh Marru, and Sun Kim. 2010. Biovlab:Bioinformatics data analysis using cloud computing and graphical workflow composers. In *Cloud Computing and Software Services: Theory and Techniques*, Syed A. Ahson and Mohammad Ilyas (Eds.). Number 309-327. CRC Press, Inc.

Jia Yu and Rajkumar Buyya. 2005. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing* 3, 3–4 (Sept. 2005), 171–200. DOI:http://dx.doi.org/10.1007/s10723-005-9010-8

Yong Zhao, Mihael Hategan, Ben Clifford, Ian Foster, Gregor von Laszewski, Veronika Nefedova, Ioan Raicu, Tiberiu Stef-Praun, and Michael Wilde. 2007. Swift: Fast, reliable, loosely coupled parallel computation. In *Proc. IEEE SERVICES'07*. IEEE Computer Society, 199–206. DOI:http://dx.doi.org/10.1109/SERVICES.2007.63

Yong Zhao, Youfu Li, Ioan Raicu, Shiyong Lu, Wenhong Tian, and Heng Liu. 2015. Enabling scalable scientific workflow management in the Cloud. *Future Gener. Comput. Syst.* 46 (2015), 3–16. DOI:http://dx.doi.org/10.1016/j.future.2014.10.023

Zhiming Zhao, Paola Grosso, Jeroen van der Ham, Ralph Koning, and Cees de Laat. 2011. An agent based network resource planner for workflow applications. *Multiagent and Grid Systems* 7, 6 (2011), 187–202.

Daniel Zinn, Quinn Hart, Timothy McPhillips, Bertram Ludäscher, Yogesh Simmhan, Michail Giakkoupis, and Viktor K. Prasanna. 2011. Towards reliable, performant workflows for streaming-applications on cloud platforms. In *Proc. IEEE/ACM CCGRID'11*. 235–244. DOI:http://dx.doi.org/10.1109/CCGrid.2011.74

Received March 2015; revised May 2016; accepted September 2016