

Analysing scientific workflows: why workflows not only connect web services

Ingo Wassink^{*||}, Paul E. van der Vet^{*||}, Katy Wolstencroft[†], Pieter B.T. Neerincx^{‡||},
Marco Roos^{§||}, Han Rauwerda^{¶||} and Timo M. Breit^{¶||}

^{*} *Human Media Interaction Group, University of Twente
Enschede, the Netherlands*

E-mail: {i.wassink, p.e.vandervet}@ewi.utwente.nl

[†] *Information Management Group, the University of Manchester
Manchester, United Kingdom*

E-mail: kwolstencroft@cs.man.ac.uk

[‡] *Laboratory of Bioinformatics, Wageningen University
Wageningen, the Netherlands*

E-mail: pieter.neerincx@wur.nl

[§] *Informatics Institute, University of Amsterdam
Amsterdam, the Netherlands*

E-mail: m.roos1@uva.nl

[¶] *MicroArray Department & Integrative Bioinformatics Unit
University of Amsterdam
Amsterdam, the Netherlands*

E-mail: {j.rauwerda, t.m.breit}@uva.nl

^{||} *the Netherlands BioInformatics Centre*

Abstract—Life science workflow systems are developed to help life scientists to conveniently connect various programs and web services. In practice however, much time is spent on data conversion, because web services provided by different organisations use different data formats. We have analysed all the Taverna workflows available at the myExperiment web site on December 11, 2008. Our analysis of the tasks in these workflows shows several noticeable aspects: their number ranges from 1 to 70 tasks per workflow; 18% of the workflows consist of a single task.

Of the tasks used are 22% web services; local services, i.e. tasks executed by the workflow system itself, are very popular and cover 57% of tasks; tasks implemented by the workflow designer, scripting tasks, are also used often (14%). Our analysis shows that over 30% of tasks are related to data conversion.

Keywords-Scientific workflow; data conversion; web services; scripting; sub-workflow

I. INTRODUCTION

A workflow system is an environment in which a scientist can describe and run in-silico experiments. Ideally, the workflow designer is not required to do any programming. Workflow systems have been developed to simplify web service discovery, web service invocation and data passing between the web services. A complication here is though that, as in any area, different organisations in the life science domain use different data formats to even represent the same

type of information [1], [2], [3], [4], [5]. This makes it difficult, if not impossible, to pass data produced by one web service to another web service without restructuring it, especially when they are hosted by different organisations. Workflow systems provide local services and scripting tasks to help scientists to connect these web services. Local services are tasks provided by and executed by the workflow system itself. These local services are used, among others, to perform string operations, to read and to write files, and to interact with users. Scripting tasks are tasks that are implemented by the workflow designer by means of scripts [6]. Scripting tasks, by contrast, are used to create tasks not available as web services and not provided by the workflow system as local services [4], to create interactive tasks [1] and to perform data transformations [5], [7]. Since different web services use different data formats, we expect that these local services and scripting tasks are often used in life science workflows to perform data transformation.

Life science workflows can be complex with respect to the number of web services, local services and scripting tasks. When workflows become more complex, hierarchy becomes important to keep the workflow comprehensible [8]. We expect large workflows to have more sub-workflows than small workflows.

In this study, we will examine the different types of tasks used in life science workflows. We will distinguish local

services, web services, scripting tasks and sub-workflows. Based on this distinction, we will establish a lower bound on the number of tasks dedicated to data conversion in life science workflows. First, we will discuss the data set used. Second we will discuss how we have analysed the data set and third, we will interpret the results. Then, we will discuss related work. We will end with a conclusion.

II. THE DATA SET

This study focuses on workflows designed in Taverna, because these workflows are easy to analyse for three reasons. First, Taverna is a very popular workflow system in the life science domain (>55.000 downloads [9]), because it is open source, provides access to many web services and offers many features that facilitate the development of (life science) workflows. Second, workflows designed in Taverna are stored using the XML language SCUFL (Simplified Conceptual Uniform Flow Language) and are therefore easy to parse and to analyse. Third, workflows designed in Taverna are easy to collect due to the existence of the myExperiment website [10]. The myExperiment website (>1.300 members [9]) enables life scientists to share their workflow design easily. Although the myExperiment website is not restricted to Taverna workflows, most of the workflows shared are Taverna workflows. The myExperiment site is online since October 2007 and at the moment of writing, it provides access to 415 Taverna workflows¹. The SCUFL files of the workflows are directly accessible through URLs. Each workflow has a unique identifier ranging from (1..N), where id=1 denotes the first workflow stored at myExperiment and id=N the last workflow stored at myExperiment.

If the SCUFL file contains sub-workflows, then these are expanded in our analysis. The tasks of the sub-workflows are counted as well as the number of sub-workflows in the SCUFL file. The graph in Figure 1 presents the number of workflows sorted by size. The number of tasks per workflow ranges from 1 to 70 tasks. The average workflow size is 8.8 tasks; the standard deviation is 11.7 tasks. The total number of tasks in all workflows together is 3660.

III. APPROACH

Ideally, we want to categorise the tasks in the workflows based on the way they are used in the workflow. Due to the huge number of tasks, manually annotating all the tasks is infeasible. To be able to tell something about the way the tasks are used, we have chosen to split the tasks into the four categories described below:

Local service: Tasks that are provided and executed by the Taverna workflow system. Examples are tasks that perform string operations, file operations and user interaction services.

¹The workflows are collected at December 11, 2008

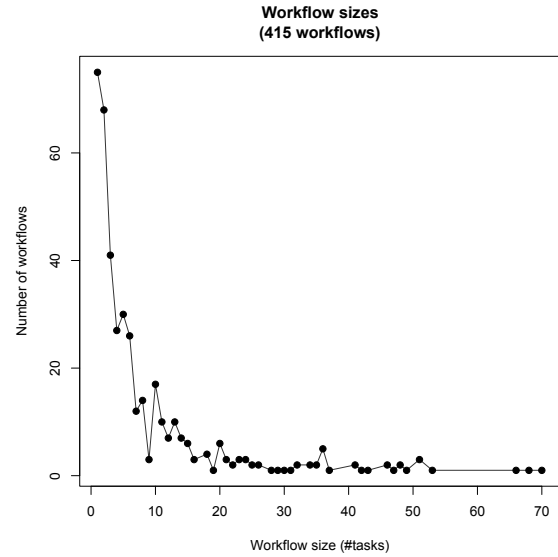


Figure 1. The distribution of the 415 Taverna workflows stored at myExperiment based on the number of tasks of the expanded workflows.

Web service: Tasks that provide access to programs hosted by third parties, on external computers (web servers). These programs can be accessed through the Internet using a protocol.

Scripting: Tasks programmed by the workflow designer. The workflow designer can insert snippets of program code and can specify the inputs the task requires and the output it produces.

Sub-workflow: These tasks are complete workflows themselves. Sub-workflows are used in large workflows to deal with complexity. Sub-workflows enable black boxing of certain parts of the workflow.

These categories are also applicable to other workflows. Based on our own experience, we expect that local services and scripting tasks are used more often for data conversion than the other categories. The analysis we performed is implemented as a workflow too (Figure 2). The workflow takes two inputs, the id of the first workflow and the id of the last workflow to be analysed. This workflow basically consists of four steps.

- 1) *Generate_URLs*; for all workflows stored at myExperiment with an id between first id and last id, the URL pointing to the location of the SCUFL file is generated. For example, the workflow with id 603 is located at <http://www.myexperiment.org/workflows/603/download>,
- 2) *Analyse_Single_workflow*; the SCUFL file of the workflow is downloaded using the Taverna task

“Get_web_page_from_URL”. The XPath task provided by Taverna is used to collect the task types used in the workflow,

- 3) *Create_R_Table*; the tasks collected by the XPath tasks are put in a table, in which each row describes the content of a single workflow. The first column contains the workflow identifier; all other columns describe the number of tasks used per task type,
- 4) *Analyse_Workflows*; the last step executes an R script we have implemented to analyse the data stored in the table and to generate the plots used in this study.

The results of the analysis task are used as the workflow outputs. The workflow is made available through the myExperiment site and can be downloaded at <http://www.myexperiment.org/workflows/648>.

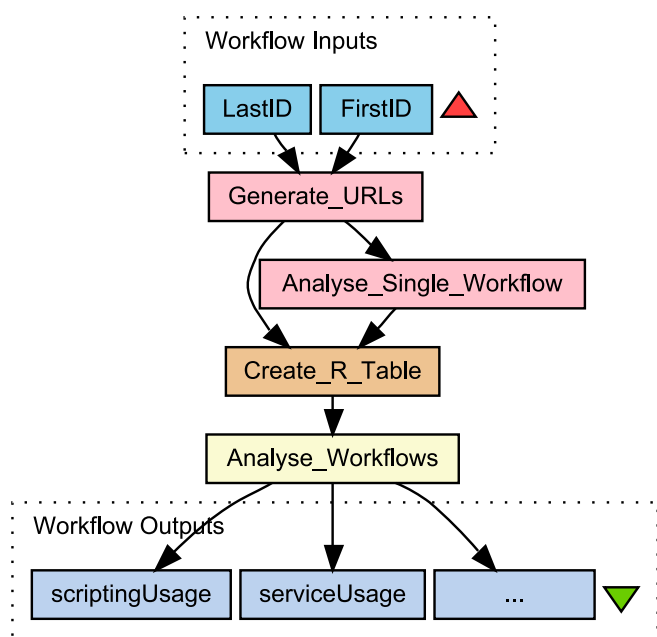


Figure 2. The workflow designed for analysing the workflows at myExperiment

IV. RESULTS

Figure 1 shows a remarkably high number of workflows that consist of a single task (75 of the 415 workflows, 18%). A closer look at the myExperiment site clarifies that many of them are used as prototypes showing how to use certain local services and web services that provide a lot of configuration parameters. Other single-task workflows are used to share scripts. Taverna provides no functionality to easily share scripting tasks, but enables its users to import workflows published at the myExperiment website directly into the Taverna workflow system. Many Taverna users apply this approach in order to reuse scripting tasks.

Figure 3 shows the service type usage in all 415 workflows. About 57% of the tasks used in the workflows are

local services and only 22% are web services. This is remarkable, since workflow systems, such as Taverna, are originally developed to connect web services. Scripting tasks count for 14% of the total number of tasks.

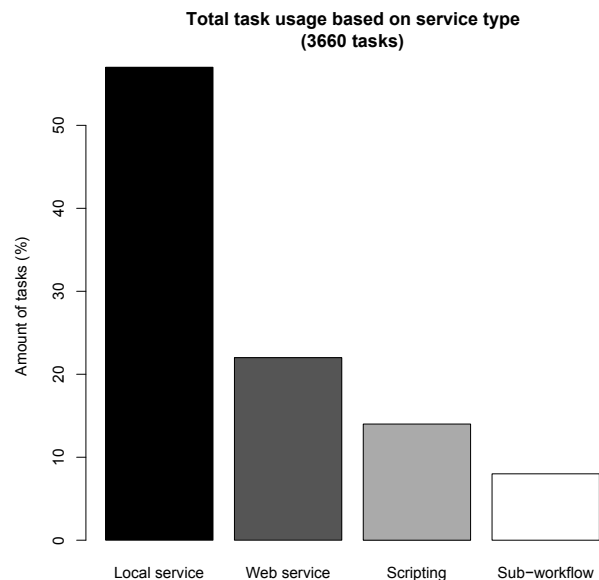


Figure 3. Task usage in all 415 workflows

The diagram in Figure 4 shows the trends of task usage when the workflow size increases. As expected, sub-workflows are seldom used in smaller workflows (workflows of size <10), but become more popular when the workflow size increases. About 8% of tasks in the workflows are sub-workflows.

The three categories local service, web service and scripting tasks will be discussed in more detail.

A. Local services

Taverna provides many local services that can be used for various tasks. Hull has developed a website describing these local services². Based on the description at this web site, we have categorised the local services on the type of operation they perform.

- CDK: These tasks represent viewers, algorithms and analysis tools provided by the Chemistry Development Kit (CDK) Java library [11], [12] and are provided by means of a plugin for Taverna by the CDK project.
- Conditional: Tasks used to construct conditional branches, such as if-then-else constructs.

²<http://www.cs.man.ac.uk/~hull/shims.html>, last visited February 11, 2009

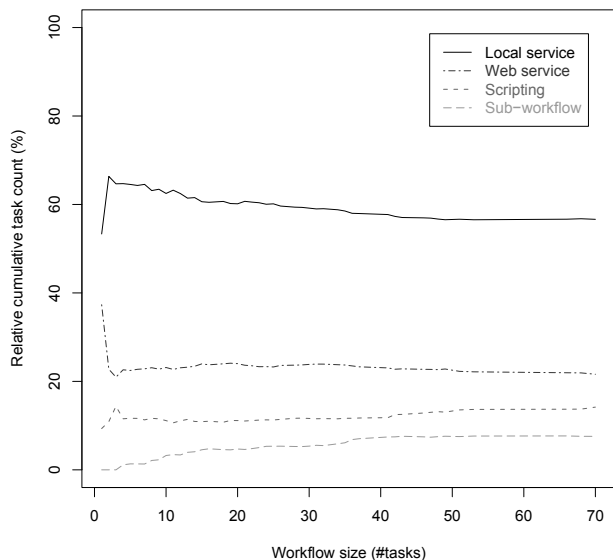


Figure 4. Trends in task usage based on service type and workflow size

- Constant:** Task that represent constant values. These are used to provide default inputs for other tasks.
- Database access:** Tasks that are part of Taverna and provide access to well-known life science databases. For many of them, web service interfaces are available, but they have been developed quite recently.
- Data conversion:** Tasks to compose, decompose or encode data without affecting the meaning of the data or their individual elements. For example, string concatenation, composing XML data objects and transforming a sequence into FASTA format.
- Operation:** Tasks that creates new biological data based on their inputs. The difference with data conversion is that the input and output are related, but do not represent the same type of information. Examples are DNA to protein translation and calculating the complement of DNA.
- Testing:** Tasks that are used to test the Taverna workflow environment, or a workflow designed in Taverna. An example is the task that generates a lot of strings.
- User interaction:** Tasks that interact with the user by means of dialog windows, such as the file selection dialog.
- Util:** General purpose task, such as file ac-

cess.

Unknown: Tasks that are not categorised, because their function is not known.

Table I shows how the local services are distributed over the different categories. Figure 5 presents the distribution of tasks among the different classes. The data conversion tasks are responsible for 53% of the local services. This means that in total 30% of all the tasks are data conversion tasks, which indicates that data incompatibility is an important problem in the life science domain.

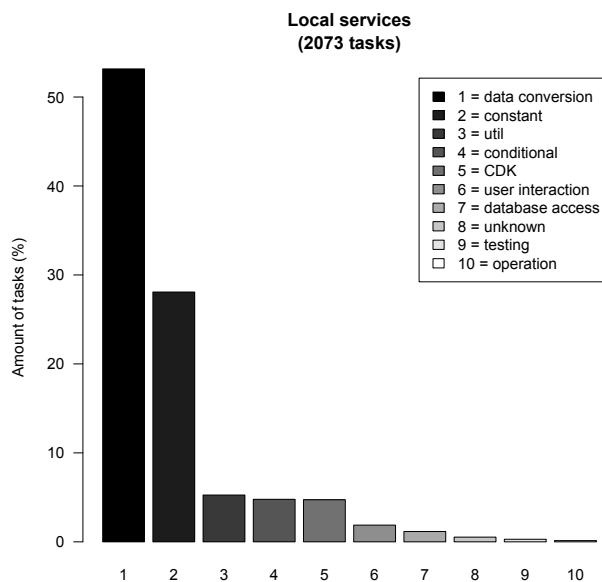


Figure 5. Use of local services, categorised by functionality

String constant tasks take a remarkable second place (28%). This is remarkable, because Taverna also enables the user to directly assign default values to input ports. The advantage of using a string constant task is that the default value is explicitly visible in the workflow. Both the conditional branching services and the util services each take 5% of the local services. The residual 5% is shared among the remaining six categories.

B. Web services

Taverna by default supports the five invocation mechanisms listed below:

SOAP/WSDL: A widely used combination of the XML based web service description language WSDL [13] and the XML based protocol SOAP [14]. SOAP/WSDL is a general web service framework, popular inside and outside the life science domain.

Table I
CATEGORISATION OF THE LOCAL SERVICES.

Class	Services		
conditional	FailIfTrue	FailIfFalse	
data conversion	GenBankParserWorker FlattenList XPathTextWorker biomobyparser StringStripDuplicates SliceList StringListMerge ByteArrayToString	XMLInputSplitter XMLOutputSplitter biomobyobject XSLTWorker StringSetIntersection EncodeBase64 StringSetDifference PadNumber	StringConcat SplitByRegex DecodeBase64 ExtractMobyData RegularExpressionStringList FilterStringList StringSetUnion
retrieval	PubMedEFetchWorker NucleotideINSDSeqXMLWorker ProteinTinySeqXMLWorker SwissProtParserWorker	ProteinINSDSeqXMLWorker ProteinGBSeqWorker NucleotideGBSeqWorker	NucleotideTinySeqXMLWorker NucleotideFastaWorker ProteinFastaWorker
operation	ReverseCompWorker	TranscribeWorker	
interaction	AskWorker TellWorker	SelectWorker WarnWorker ChooseWorker	
util	FileListByRegexTask ConcatenateFileListWorker TextFileReader LocalCommand	EchoList FileListByExtTask WebPageFetcher ExtractImageLinks	WebImageFetcher apiconsumer TextFileWriter SQLQueryWorker
constant	stringconstant		
testing	EmitLotsOfStrings	TestSometimesFails	TestAlwaysFailingProcessor
unknown	alternate helpurl	knowarcjanitor	arbitrarygt4

SoapLab This framework [15]: is developed by EBI to easily encapsulate existing bioinformatics algorithms and tools as web services.

SeqHound: The SeqHound services [16] provide access to biological sequence and structure databases.

BioMart: This [16] is a query-oriented data management system that provides access to, among others, the Ensembl [12] database.

MOBY-S: This [17], [18], [19] is a branch of the BioMOBY consortium that simplifies the discovery process of life science services and the access to these web services by maintaining a central registry.

Figure 6 shows the use of the different invocation mechanisms in Taverna. The SOAP/WSDL web services are by far most popular (~66%). SoapLab web services take a second place (~24%). MOBY-S, BioMart and SeqHound web services are used to a lesser extent in Taverna. An explanation for this is the existence of other powerful applications that provide access to these web services too. BioMart web services are accessible through the MartView web applica-

tion; MOBY-S web services are accessible through various tools, such as Seahawk [20]. These tools often provide better facilities to search for web services and to connect them. Taverna becomes the preferred application when these web services are embedded in experiments that need also access to web services using other invocation mechanisms.

Another reason is that SOAP/WSDL web services and SoapLab web services support a greater number of functions. For example, BioMart provides database access. Once data is collected from a database using a BioMart web service, there are plenty of things one can do to analyse the data, using numerous SOAP/WSDL web services and SoapLab web services.

C. Scripting

Taverna supports the two programming languages Java and R [21] through respectively the BeanShell processor and the RShell processor [6]³. Java provides many facilities to handle data and to create advanced used interfaces and therefore is useful to implement a wide variety of tasks. The BeanShell processor is very popular, 97% of the scripting tasks are implemented using this one. In the workflows

³The Taverna term *processor* refers to what we call a task

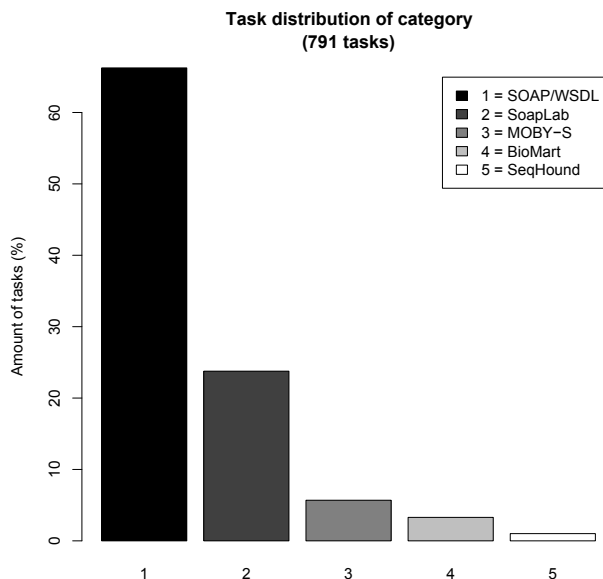


Figure 6. Different types of invocation mechanism used in the workflows

stored at myExperiment, the BeanShell processor is used among others to perform data conversion and to create interactive tasks.

The R language is specifically designed to perform statistical analysis. In Taverna, all the RShell processors (3% of the scripting tasks) are used to perform statistical analyses and to create plots.

V. RELATED WORK

Lord et al. [22] have investigated the bioinformatics web services available in terms of invocation mechanism. They found that 25% of the services available are SoapLab services, 30% are MOBY-S services and 30% are REST services. Only 5% of the bioinformatics services available are SOAP/WSDL services. This indicates that the use of services differs significantly from the availability of services. Lord et al. note that 10% of the services available are provided as workflows. This closely matches the 8% sub-workflows used in Taverna we measured.

Workflow systems should form the ideal tool with which scientists can easily connect different web services to compose an in-silico experiment. In practice, however, dealing with data incompatibility problems forms a large part of the workflow designers' daily activities [3], [23]. The problem of data incompatibility is recognised in the life science domain [7], [24]. Both Taverna and Kepler provide so-called *shim* services. These are "helper" functions to transform closely related data without performing any scientific operation and can be local services, scripting tasks, and even web services [25]. The MOBY-S plugin [18] for Taverna

provides so-called composer and splitter tasks to compose and decompose MOBY-S objects. In similar fashion, Taverna has composer and splitter tasks to compose and decompose XML data used by SOAP/WSDL services. The size of the problem is, however, never investigated before.

VI. DISCUSSION & CONCLUSION

Scientific workflow systems have been developed to simplify the construction of in-silico experiments by providing a graphical user interface by which end-users can orchestrate web services into complex workflows [9], [26]. However, currently workflow designers spend most of their time on data-conversion [23], which our study confirms. About 30% of the tasks in these workflows represent data-conversion activities. This is just a lower limit. There are three reasons why the real amount of data conversion tasks is even higher.

- 1) A lot of BeanShell processors are used to perform data conversion.
- 2) Some web services are implemented to perform data transformation. Not all scientists know how to program in the languages supported by Taverna. These scientists prefer to program in a language they already know, such as Perl. For them, it is easier to implement a new web service and to provide access to this service through SoapLab or MOBY-S. This newly created data conversion service is directly accessible through Taverna, even for other people. The MOBY-S central registry has a special category for these web services called "Conversion" web services. At the moment of writing, this category contains 65 services, of which only a few are used in the workflows we studied.
- 3) Some services are not intended to perform data conversion, however, some users use them for this purpose. For example, scientists send multiple requests that only differ in the output format requested to the same database. In such a case, the database web service is used to perform data conversion rather than collecting data.

The analysis is restricted to Taverna workflows, but probably can be generalised to other workflow systems and even to scientific programming. The problems with data incompatibility can only be solved if different organisations agree on standards [24]. However, as long as people are free to define their own standard, they will.

Although its primary goal is connecting local and web services [27], [28], [29], [9], a scientific workflow system should not be seen as just a web service composition environment. It is a visual programming environment that aims to reduce the programming effort required by the scientists [8]. A workflow system has its advantages with respect to programming languages. It can guide people to perform data conversion, for example, by providing tasks to compose and decompose complex data structures. If the required tasks are not available as a local service or web

service, the workflow system can help the workflow designer to implement these tasks by means of scripting tasks for programming languages the designer is familiar with. Web sites for sharing workflows, such as myExperiment, form a valuable source for dealing with data incompatibility and prevent workflow designers to develop these tasks themselves. A deeper analysis of the data conversion tasks currently used can help to find out what kind of data conversions are required and how a workflow system can help its users to perform them.

ACKNOWLEDGMENT

This work was part of the BioRange programme of the Netherlands Bioinformatics Centre (NBIC), which is supported by a BSIK grant through the Netherlands Genomics Initiative (NGI).

REFERENCES

- [1] C. Wroe, C. Goble, M. Greenwood, P. Lord, S. Miles, J. Pappay, T. Payne, and L. Moreau, "Automating experiments using semantic data on a bioinformatics grid," *IEEE Intelligent Systems*, vol. 19, no. 1, pp. 48–55, 2004.
- [2] P. Neerincx and J. Leunissen, "Evolution of web services in bioinformatics," *Briefings in Bioinformatics*, vol. 6, no. 2, pp. 178–188, 2005.
- [3] K. Belhajjame, S. Embury, and N. Paton, "On characterising and identifying mismatches in scientific workflows," in *Data Integration in the Life Sciences (DILS'06)*, S. Istrail, P. Pevzner, and M. Waterman, Eds., Hinxton, UK, 2006, pp. 240–247.
- [4] M. Kappler, "Software for rapid prototyping in the pharmaceutical and biotechnology industries," *Current Opinion in Drug Discovery & Development*, vol. 11, no. 3, pp. 389–392, 2008.
- [5] J. Shon, H. Ohkawa, and J. Hammer, "Scientific workflows as productivity tools for drug discovery," *Current Opinion in Drug Discovery & Development*, vol. 11, no. 3, pp. 381–388, 2008.
- [6] L. Peter, J. Castrillo, G. Velarde, I. Wassink, S. Soiland-Reyes, S. Owen, D. Withers, T. Oinn, M. Pocock, C. Goble, S. Oliver, and D. Kell, "Performing statistical analyses on quantitative data in Taverna workflows: an example using R and maxdBrowse to identify differentially-expressed genes from microarray data," *BMC Bioinformatics*, vol. 9, no. 334, pp. 1–38, 2008.
- [7] S. Bowers and B. Ludäscher, *Data Integration in the Life Sciences*. Berlin, DE: Springer Verlag, 2004, ch. An Ontology-Driven Framework for Data Transformation in Scientific Workflows, pp. 1–16.
- [8] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. In press, 2008.
- [9] D. De Roure and C. Goble, "Software design for empowering scientists," *Software, IEEE*, vol. 26, no. 1, pp. 88–95, 2009.
- [10] C. Goble and D. De Roure, "myExperiment: Social networking for workflow-using e-scientists," in *WORKS'07*, E. Deelman and I. Taylor, Eds., Monterey, California, USA., 2007, p. Online.
- [11] C. Steinbeck, Y. Q. Han, S. Kuhn, O. Horlacher, E. Luttmann, and E. Willighagen, "The Chemistry Development Kit (CDK): An open-source Java library for chemo- and bioinformatics," *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 2, pp. 493–500, 2003.
- [12] C. Steinbeck, C. Hoppe, S. Kuhn, M. Floris, R. Guha, and E. Willighagen, "Recent developments of the Chemistry Development Kit (CDK) - an open-source Java library for chemo- and bioinformatics," *Current pharmaceutical design*, vol. 12, no. 17, pp. 2111–2120, 2006.
- [13] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "WSDL," World Wide Web electronic publication, W3C, Tech. Rep. NOTE-wsdl-20010315, 2001. [Online]. Available: <http://www.w3.org/TR/wsdl>
- [14] D. Box, E. D., K. G., A. Layman, M. N., H. Frystyk Nielsen, S. Thatte, and D. Winer, "SOAP," World Wide Web electronic publication, W3C, Tech. Rep. NOTE-SOAP-20000508, 2000. [Online]. Available: <http://www.w3.org/TR/soap/>
- [15] M. Senger, P. Rice, and T. Oinn, "SoapLab - a unified sesame door to analysis tools proceedings," in *Proceedings of the UK e-Science- All Hands Meeting 2003*, S. Cox, Ed., Nottingham, UK, 2003, pp. 509–513.
- [16] S. Durinck, Y. Moreau, A. Kasprzyk, S. Davis, B. De Moor, A. Brazma, and W. Huber, "BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis," *Bioinformatics*, vol. 21, no. 16, pp. 3439–3440, 2005.
- [17] M. D. Wilkinson and M. Links, "BioMOBY: an open source biological web services proposal," *Briefings in Bioinformatics*, vol. 3, no. 4, pp. 331–341, December 2002.
- [18] E. Kawas, M. Senger, and M. D. Wilkinson, "BioMoby extensions to the Taverna workflow management and enactment software," *BMC Bioinformatics*, vol. 7, no. 523, pp. 1–13, 2006.
- [19] The BioMoby Consortium, "Interoperability with Moby 1.0—It's better than sharing your toothbrush!" *Briefings in Bioinformatics*, vol. 9, no. 3, pp. 220–231, 2008.
- [20] P. Gordon and C. Sensen, "Seahawk: moving beyond html in web-based bioinformatics analysis," *BMC bioinformatics*, vol. 8, no. 208, pp. 1–13, 2007.
- [21] R. Ihaka and R. Gentleman, "R: A language for data analysis and graphics," *Journal of Computational and Graphical Statistics*, vol. 5, no. 3, pp. 299–314, 1996.
- [22] P. Lord, P. Alper, C. Wroe, and C. Goble, "Feta: A lightweight architecture for user oriented semantic service discovery," in *The Semantic Web: Research and Applications*, A. Gómez-Pérez and J. Euzenat, Eds., Crete, Greece, 2005, pp. 17–31.

- [23] P. Gordon and C. Sensen, "A pilot study into the usability of a scientific workflow construction tool," Sun Center of Excellence for Visual Genomics, Tech. Rep. 2007-874-26, 2007.
- [24] P. N. Seibel, J. Krüger, S. Hartmeier, K. Schwarzer, K. Löwenthal, H. Mersch, T. Dandekar, and R. Giegerich, "XML schemas for common bioinformatic data types and their application in workflow systems," *BMC Bioinformatics*, vol. 7, no. 490, pp. 1–11, 2006.
- [25] D. Hull, R. Stevens, P. Lord, C. Wroe, and C. Goble, "Treating semantic web syndrome with ontologies," in *First AKT workshop on Semantic Web Services (AKT-SWS04) KMi*, J. Domingue, L. Cabral, and E. Motta, Eds., The Open University, Milton Keynes, UK., 2004, pp. 1–4, workshop proceedings CEUR-WS.org ISSN:1613-0073.
- [26] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers, "Examining the challenges of scientific workflows," *Computer*, vol. 40, no. 12, pp. 24–32, 2007.
- [27] M. Addis, J. Ferris, M. Greenwood, P. Li, D. Marvin, T. Oinn, and A. Wipat, "Experiences with e-Science workflow specification and enactment in bioinformatics," in *e-Science All Hands Meeting 2003*, S. Cox, Ed., Nottingham, United Kingdom, 2004, pp. 459–466.
- [28] A. Barker and J. Van Hemert, "Scientific workflow: A survey and research directions," in *Parallel Processing and Applied Mathematics*, R. Wyrzykowski, J. Dongarra, K. Karczewski, and JerzyWasniewski, Eds., Gdansk, Poland, 2007, pp. 746–753.
- [29] A. Gibson, M. Gamble, K. Wolstencroft, T. Oinn, and C. Goble, "The data playground: An intuitive workflow specification environment," in *E-SCIENCE '07: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*, S. Cox, Ed. Washington, DC, USA: IEEE Computer Society Press, 2007, pp. 59–68.