WORKFLOW RE-USE AND DISCOVERY IN BIOINFORMATICS

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2008

By
Antoon Goderis
School of Computer Science

Contents

List of Tables

List of Figures

Abstract

Scientists in many disciplines are increasingly faced with analysing a deluge of scientific data from sources scattered across the globe. Workflow techniques have the potential to become an important part of on-line experimentation as they allow scientists to describe and enact their experimental processes in a structured, repeatable and verifiable way.

Given the availability of scientist-friendly workflow editors, scientists are moving away from cutting and pasting data between Web pages in favour of producing automated workflows based on Web services. An increasingly large pool of workflows is being shared and made available for re-use. The notion that these workflows and the experimental processes they represent are a useful, re-usable artifact in their own right is new. As a new phenomenon, scientific workflow re-use and discovery is not well understood and it is unclear whether and how it could be supported automatically.

The thesis analyses the workflow re-use and discovery process based on surveys, interviews and user experiments with scientists and scientific programmers from different disciplines. We also analyse the impact of using multiple models of computation on workflow re-use. In particular, we show how some models of computation are better re-usable than others.

Further, we capture and model scientist re-use and discovery behaviour when re-using data flow workflows from the bioinformatics domain. The result is a suite of human benchmarks of value to developers of workflow discovery techniques.

Finally, the benchmarks enable us to evaluate a range of existing service discovery based techniques and novel workflow-structure based discovery techniques. The techniques vary in the language they work over (natural language or a Semantic Web language) and the level of workflow detail they process. The evaluation shows that performance of the workflow discovery techniques swings substantially depending on the task in question. This argues in favour of a multi-varied approach that combines multiple techniques.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Copyright

Acknowledgements

I would like to express a sincere thank you to Carole for having taken the bet with Mark and for being all that she is. Also many thanks to advisor Alan Rector, external consultant Uli Sattler, my collaborators, dear friends, my parents, loving family and colleagues inside and outside of the Information Management Group. I gratefully acknowledge the support from EPSRC and WUN.

Chapter 1

Introduction

As more scientific resources become available on the World Wide Web, scientists increasingly rely on Web technology for performing *in silico* (*i.e.* computerised) experiments. With the publication of scientific resources as services on the Web or as services on the computational Grid or Cloud, scientists are making a shift from traditionally copying and pasting their data through a sequence of Web pages offering those resources, to the creation and use of distributed processes for experiment design, data analysis and knowledge discovery. Research councils in various countries have set out to build a global infrastructure to support this under the banner of *e-Science*. e-Science translates the notion of virtual organisations into a customised Grid or Cloud middleware layer for scientists, thereby aiming to increase collaboration within and between scientific fields [HT02][GDE+07]. Workflow techniques are an important part of *in silico* experimentation, potentially allowing the e-Scientist to describe and enact their experimental processes in a structured, repeatable and verifiable way.

For example, the ^{my}Grid project[1] has produced the open source Taverna workflow editor[2] to build workflows in bioinformatics. Taverna provides access to hundreds of distributed services which offer in excess of thirteen hundred operations. To date scientists have produced over five hundred workflows with Taverna, some of which orchestrate up to fifty distributed services. These resources have been developed by users and service providers distributed throughout the global biology community. Figure 1.1 shows an example of a Taverna-made workflow which gathers information about genetic sequences in support of research on Williams Beuren syndrome (WBS) [STW+04]. The diagram shows a bioinformatics pipeline, producing lots of

[1] Web site: `www.mygrid.org.uk`
[2] Web site: `taverna.sf.net`

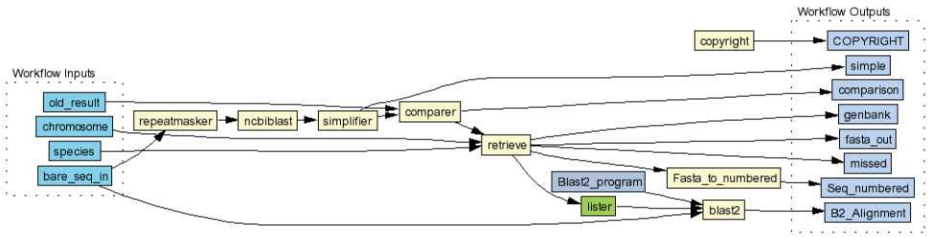


Figure 1.1: Part of a Taverna workflow to annotate genetic sequences.

data from a limited number of inputs (the left and right boxes) based on a set of autonomous distributed services.

We are now witnessing how scientists have started re-using and propagating *in silico* experiments as commodities and "know-how" in their own right. To cater for the re-use and discovery of *in silico* experiments on the scale of the Web, the e-Science infrastructure will need to expand its current handling of the workflow life cycle. An early example of such an infrastructure is the Web-based collaboration and sharing platform developed in the ^{my}Experiment project.[3]We see workflow re-use and discovery as a way of bootstrapping a "Web of Science" by stimulating the dynamics of sharing and re-using experimental components in the scientific community.

1.1 Research questions

The key question asked in this thesis is:

What is scientific workflow re-use and discovery and can we support it automatically?

Four research questions were pursued to address this central question:

Q1 What requirements should be fulfilled for workflow re-use and discovery to occur in science?

To be able to support the process of scientific workflow re-use and the subproblem of discovering relevant workflows, we need to understand what drives the process. We wish to determine the necessary conditions for re-use and discovery.

[3]Web site: `www.myexperiment.org`

Q2 How do scientists re-use and discover workflows?

Workflow re-use and discovery is a process performed by scientists. By observing how they go about the process and by capturing their behaviour, we gain additional insight into the process.

Q3 Can automated discovery techniques support workflow re-use and discovery?

Once an understanding of the process of workflow re-use and discovery is available, we can consider the design and use of automated techniques to support it. Whether the techniques are effective should be assessed by measuring their performance on tasks usually performed manually by scientists.

Q4 How does having multiple models of computation in a workflow affect workflow re-use?

Workflows are computational artifacts rooted in a particular model of computation. **Q1 - Q3** assume that a single model of computation exists, in accordance with the current state of the art. However, depending on how scientists model an *in silico* analysis, they may choose and combine different models of computation. The choice of model of computation affects the re-usability of a workflow, but its impact is currently not well understood.

1.2 Thesis structure

The structure of the thesis reflects the four research questions. They are addressed in order.

Chapter 2 defines the problem of workflow re-use and repurposing. It explains the phenomenon of workflows in science and presents a requirements analysis for workflow re-use drawn from surveys and interviews with scientists and scientific programmers.

Chapter 3 defines and scopes the workflow discovery problem, presents technical requirements and categorises related work on discovery techniques.

Chapter 4 reports on a series of user experiments set up to capture and model behaviour during re-use and discovery for one class of scientists, namely bioinformaticians. It leads to a series of human-made benchmarks.

Chapter 5 presents the range of automated techniques we considered to support the re-use and discovery problem. The techniques vary in the level of detail at which they

process workflow information and the formalism they expect this information to be in, either natural language or annotation in a Semantic Web language.

Chapter 6 evaluates to what extent the automated techniques support the re-use and discovery of workflows by bioinformaticians. The benchmarks of Chapter 4 are central to their assessment.

Chapter 7 investigates the final research question by analysing the link between multiple models of computation. A classification is built showing valid re-use cases between workflows that adhere to different models of computation.

Chapter 8 concludes and sets out avenues for future research.

1.3 Publications

Some of the material in the thesis is based on previous publications.

The requirements analysis in Chapters 2 and 3 is based on the following papers and book chapters. With the exception of Ulrike Sattler, all co-authors are collaborators from the ^{my}Grid project.

- Antoon Goderis, Ulrike Sattler, Phillip Lord, and Carole Goble. *Seven bottlenecks to workflow reuse and repurposing.* In Int. Semantic Web Conference ISWC05, volume 3792, pages 323337, Galway, Ireland, 2005.

- Chris Wroe, Carole Goble, Antoon Goderis, Phillip Lord, Simon Miles, Juri Papay, Pinar Alper, and Luc Moreau. *Recycling workflows and services through discovery and reuse*. Research articles. Concurr. Comput. : Pract. Exper., 19(2):181194, 2007.

- Carole Goble, Katy Wolstencroft, Antoon Goderis, Duncan Hull, Jun Zhao, Pinar Alper, Phillip Lord, Chris Wroe, Khalid Belhajjame, Daniele Turi, Robert Stevens, Tom Oinn, and David De Roure. Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences, chapter *Knowledge Discovery for Biology with Taverna: Producing and Consuming Semantics in the Web of Science.* 2006.

- Tom Oinn, Mark Greenwood, Matthew Addis, Nedim Alpdemir, Justin Ferris, Kevin Glover, Carole Goble, Antoon Goderis, Duncan Hull, Darren Marvin, Peter Li, Phillip Lord, Matthew Pocock, Martin Senger, Robert Stevens, Anil

Wipat, and Chris Wroe. *Taverna: Lessons in creating a workflow environment for the life sciences.* Concurrency and Computation: Practice and Experience: Special Issue on Scientific Workflows, 2005.

- Tom Oinn, Peter Li, Douglas B. Kell, Carole Goble, Antoon Goderis, Mark Greenwood, Duncan Hull, Robert Stevens, Daniele Turi, and Jun Zhao. Workflows for e-Science: scientific workflows for Grids, chapter *Taverna/myGrid: Aligning a Workflow System with the Life Sciences Community*, pages 300-319. Springer, 2007.

A user survey reported on in Chapter 2, one of the small user experiments in Chapter 4 and a graph-matching based technique presented in Chapter 5 were published together with Peter Li and Carole Goble.

- Antoon Goderis, Peter Li, and Carole Goble. *Workflow discovery: the problem, a case study from e-science and a graph-based solution.* In IEEE Int. Conf. on Web Services, Chicago, USA, September 18-22 2006.

- Antoon Goderis, Peter Li, and Carole Goble. *Workflow discovery: requirements from escience and a graph-based solution.* International Journal of Web Services Research (JWSR), Accepted for publication.

Work on the suitability of the OWL language for workflow discovery, joint with Ulrike Sattler and Carole Goble, is presented in Chapter 5 and published as below.

- Antoon Goderis, Ulrike Sattler, and Carole Goble. *Applying descriptions logics for workflow reuse and repurposing.* In International Description Logics Workshop, Edinburgh, Scotland, 2005.

Finally, the material in Chapter 7 is the result of a collaboration with Christopher Brooks and Edward E. Lee from the Ptolemy II project, Ilkay Altintas from the Kepler project and Carole Goble.

- Antoon Goderis, Christopher Brooks, Ilkay Altintas, Edward E. Lee, and Carole Goble. *Composing different models of computation in Kepler and Ptolemy II.* In Proc. of the 2nd International Workshop on Workflow Systems in e-Science (WSES 07) in conjunction with the International Conference on Computational Science, Beijing, China, May 27-30 2007.

- Antoon Goderis, Christopher Brooks, Ilkay Altintas, Edward A. Lee, and Carole Goble. *Composing heterogeneous models of computation in Kepler and Ptolemy II.* Future Generation Computer Systems (FGCS), Accepted for publication.

1.4 External contributions

This thesis is the original work of the author except for the following.

In Chapter 3, the formal definition of the matching conditions and the workflow discovery tasks was developed in collaboration with Khalid Belhajjame.

In Chapter 5, regarding the Description Logics section, Ulrike Sattler was instrumental for the technical analysis.

In Chapter 7, Edward A. Lee did the bulk of the technical analysis.

Chapter 2

Workflows, workflow re-use and repurposing

Throughout the thesis, we refer to scientific workflows, workflow re-use and workflow repurposing. The chapter clarifies what we mean by these terms. There are two major sections to the chapter:

- *Workflows in science.* This motivates the use of workflows by scientists, analyses the anatomy of a scientific workflow and provides the formal definition of a workflow used throughout the thesis.
- *Workflow re-use and repurposing.* Practical case studies of workflow re-use are presented from several disciplines. They lead to a set of requirements for workflow re-use and making the distinction between re-use and repurposing.

2.1 Workflows in science

Scientists are facing new challenges, which leads them to the use of workflows. In this section, we clarify their motives, analyse what defines a scientific workflow and adopt a formal definition of a workflow for our purposes.

2.1.1 Why workflows in science?

The concept of a workflow is making inroads in science due to its central role as the "glue" to connect data management, analysis, simulation and visualisation services over often voluminous and structurally and semantically complex, distributed scientific

data and services [GL05]. Special issues of journals[1] and dedicated workshops report on the state of the art of workflow systems for "e-science."[2]

Case study: workflows in bioinformatics

To make the concept of a scientific workflow more concrete, let us consider the impact of a workflow on the activities of one type of scientist: the bioinformatician. Bioinformatics is but one discipline of science. The advantages identified below hold across disciplines, however.

Analyses of data are undertaken in bioinformatics in order to test a hypothesis, derive a summary or search for patterns [STW+04]. These procedures involve the use of local and remote resources which may be information repositories such as the EMBL[3] and Swiss-Prot[4] databases, or computational analysis tools like BLAST[5] and ClustalW.[6] Such procedures are workflows where the flow of data between resources has been directed in a pre-defined manner [OGA+05].

The increasing accessibility of these resources as Web Services in addition to the availability of workflow technology has enabled bioinformatics scientists to explicitly define how data analyses should be executed as scripts which can then be stored for later use and shared within the life sciences community [STW+04] [LHJ+04] [FHW+07]. Figure 2.1 shows the example of a workflow to retrieve genetic sequence based on an identifier associated with a microarray experiment. The pane on the right shows the third-party services available to construct the workflow with.

In a survey with 24 bioinformaticians from 19 different research laboratories (see Appendix A for the list of participants), we found that workflows have been found applicable and built for almost every area of bioinformatics. The majority of participants use workflows for sequence analysis (61.9%), genome annotation (42.9%), comparative genomics (23.8%) and analysis of gene expression (23.8%). Other applications include modelling biological systems (14.3%), computational evolutionary biology (14.3%) and protein-protein docking (9.5%). Other mentioned areas were the

[1] See SIGMOD Record (2005), Journal of Grid Computing (2005), Scientific Programming (2006), Concurrency and Compuration: Practices and Experience (2006) and Future Generation Computer Systems (2008).

[2] See the NSF Workshop on the Challenges of Scientific Workflows (2006)[GRD+07], WWWF at GridAsia2007 and the Workflow Systems for E-Science workshop series (2005-2007).

[3] Web site: `www.ebi.ac.uk/embl`

[4] Web site: `www.ebi.ac.uk/swissprot`

[5] Web site: `www.ncbi.nlm.nih.gov/blast`

[6] Web site: `www.ebi.ac.uk/clustalw`

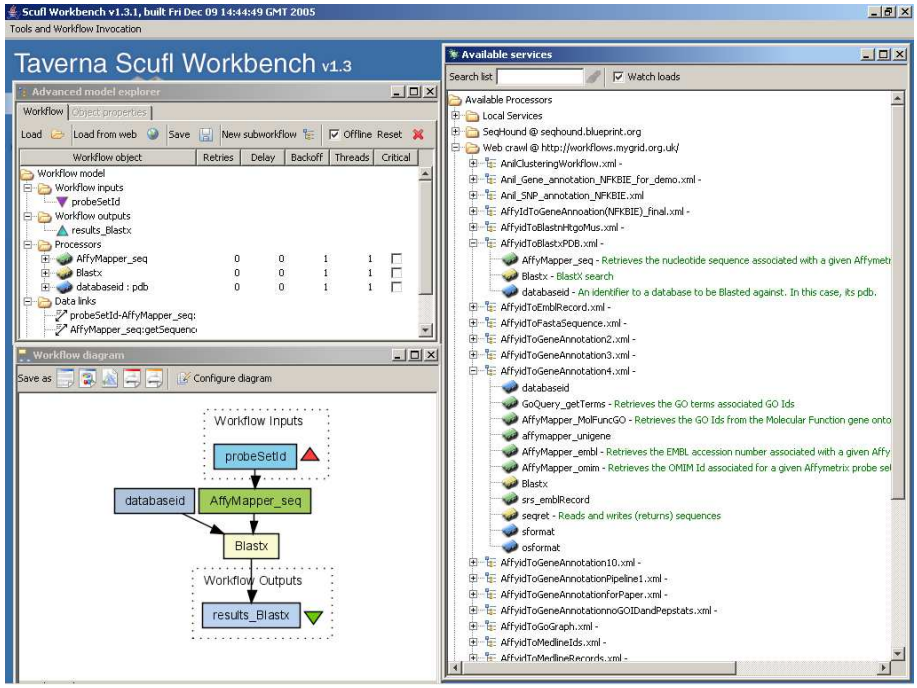


Figure 2.1: A bioinformatics workflow loaded in the Taverna workbench.

measurement of biodiversity, analysis of regulation, analysis of protein expression, analysis of mutations in cancer, prediction of protein structure and text mining.

Advantages of workflows

There are a number of advantages associated with using workflow technology for the analysis of scientific data and for bioinformatics data specifically, as opposed to the traditional manually cutting and pasting of data between forms in Web browsers.

1) Automation. The increase in the volume of data year upon year due to the continuous generation of new primary and derived data is well recognised [HT02]. Automated analyses of these data is essential since manual analysis cannot keep up with the pace of its generation. In addition, the data to be analysed require computer processing power exceeding the capabilities of an individual scientist's workstation. This leads to increasing demands for services which offer easy access to compute power. Workflow systems can *help with the high throughput analysis of data*. Firstly, workflows can automate the flow of data between analysis services. Secondly, stored workflows can be enacted on demand so that the latest data can be used in the analysis as well as new input data.

2) Fault tolerance. Issues with the reliability of services are particularly prevalent in the area of bioinformatics [LPA06]. Academic and non-commercial organisations deploy Web services for public use by scientists in the life sciences community without any prior service level agreements. Such services are used by scientists knowing of their unreliability despite the fact that they may not always be available [STW+04]. It is therefore essential that such scientific workflows are executed using the most reliable of services especially if workflows are long lived during their execution. Workflow systems potentially offer *sophisticated caching, retry and fail-over mechanisms*.

3) Recording of provenance. The building of reliable workflows is reliant upon data in the form of performance metrics about services which may be captured during the enactment of workflows. This information is provenance which identifies the source and processing of data by recording the metadata and intermediate results associated with the workflow. This type of information provides a useful audit trail since its analysis can determine the origin of erroneous or unexpected results which can sometimes be produced by workflow processes. Provenance information covers other types of information about the workflow too. For example about who built the workflow, why it was built, when it was last run, what the results were etc. See e.g. [ZWG+04] for an overview. Workflow systems potentially offer an infrastructure to *manage provenance information*.

4) Best practice. Workflows can be complex especially if many services are used in the analysis pipeline. The construction of such complex workflows requires substantial intellectual effort since intimate knowledge of the services used in the workflows is required from a syntactic and semantic standpoint for connecting the flow of data between services in a meaningful manner. In addition, analysis tools such as Blast and ClustalW are sophisticated services requiring extensive parameterisation in their use. Workflows explicitly *capture the values of* these *parameters as well as other metadata used by the workflow author* showing its best practice which can then be shared within the community.

The advantages need to outweigh a number of disadvantages in order for scientists to adopt workflows. Workflows come at a price: scientists need to acquire the skill set to design workflows and to manage larger data sets. The work in the thesis aims to support scientists in the design task by examining the potential for workflow re-use and discovery.

2.1.2 Anatomy of a scientific workflow

Workflows come in many varieties. To clarify which class of workflow we target in this thesis, we provide a brief overview of the scientific workflow landscape.

Business versus scientific workflows

What is a scientific workflow? In an introduction to a special section of SIGMOD Record on Scientific Workflows [GL05], Ludaescher and Goble introduce the concept by contrasting scientific workflows with business-oriented workflows. They write that scientific workflows are:

- typically *data-centric, dataflow-oriented* "analysis pipelines," as opposed to task-centric and control-flow oriented business workflows;
- potentially very *computationally expensive*;
- *metadata and annotation-intensive*, since the repurposing of a scientific data product in another scientist's study requires detailed, and preferably machine-processable, context and data provenance information;
- written by scientists, who are rather *individualistic* and are more likely to create their own "knowledge discovery workflows," whereas in business users are commonly restricted to using carefully designed and predetermined workflows in a constrained way.

Within the setting of supporting science, there is a lot of variety in workflow systems [WGG+07]. They vary in:

- the workflow languages they use;
- the kinds of domain and scientific process they represent;
- workflow deployment and execution environments;
- the tools and mechanisms for supporting workflow composition;
- the granularity of the services they orchestrate; and
- the way their workflows are used.

A comprehensive analysis of different scientific workflow environments is outside of our scope. See [YB05] for a survey. We limit ourselves to a discussion of the diversity found in workflow languages, which is lacking in [YB05]. We discuss two factors which contribute to the diversity: (i) the variety in levels of abstraction that workflow systems support and (ii) the models of computation they choose to govern the interaction between services in a workflow.

Levels of abstraction

Scientific workflows are described, configured and managed at different levels of abstraction to support the different phases in their lifecycle. The workflow lifecycle entails the following phases:

1. During design, while the workflow is still being designed.
2. Post design, pre-enactment, as either a finished, concrete workflow where the required resources are known, or as a finished yet abstract workflow (also known as a template) whose resources still will need to be decided dynamically during enactment.
3. During enactment, when intermediary results come about.
4. Post enactment, when all results are available.

Several workflow representations exist to support these four phases. One of the more elaborate formal schemes to describe a scientific workflow is proposed by the KWf-Grid framework [NHG06]. A workflow is described in terms of five levels of abstraction, corresponding to increasingly concrete workflows: (i) User request, (ii) Abstract workflow, (iii) Service candidates, (iv) Service instances and (v) Grid resources. The levels are shown in Figure 2.2.[7]

The KWf-Grid framework focuses on describing workflows during the design and enactment phases of the workflow lifecycle. One of the more elaborate formal schemes to describe and relate the results of workflows post enactment is proposed by the PASOA framework for managing provenance information [Gro07a].

Metadata descriptions can be associated with the workflow in each phase to facilitate retrieval and enactment. Workflow systems differ on the levels of abstraction they focus on, how the levels are represented and how the levels interact. Consequently, their workflow languages are targeted at supporting different levels of abstraction. In the mentioned KWfGrid, each of the abstraction levels uses the same workflow description language, so it is possible to mix several levels of abstraction in one workflow. Most workflow languages however concentrate on specific levels. Some focus on the specification of abstract workflows and service candidates (e.g. WINGS [GDE+07]), others on orchestrating service instances (e.g. Taverna's Scufl language [OGA+05] or BPEL[8] (the Business Process Execution Language)); still others on scheduling Grid resources (e.g. DAGMan[9]). Provenance management systems concentrate on managing

[7] Origin: the KWfGrid Web site at `www.gridworkflow.org/kwfgrid/gwes/docs/`
[8] Web site: `www.oasis-open.org/committees/wsbpel`
[9] Web site: `www.cs.wisc.edu/condor/dagman`

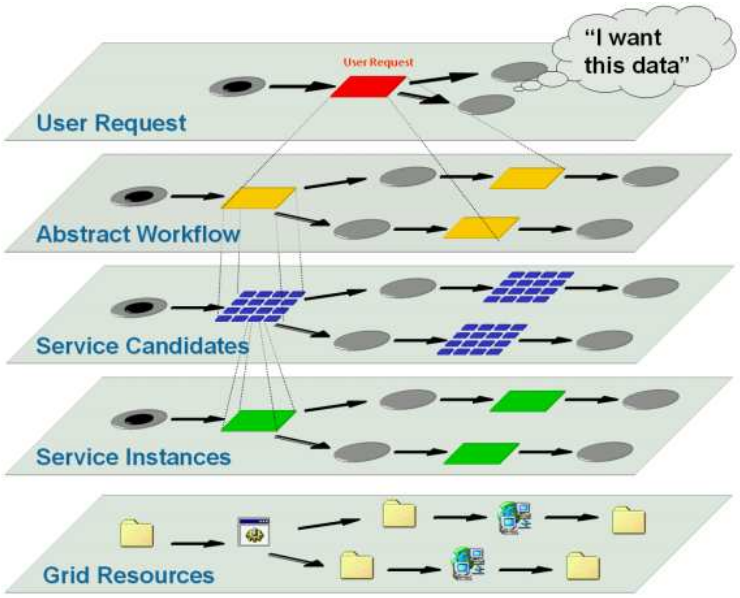


Figure 2.2: Five workflow layers of abstraction.

workflow results (see [Zha07] for an overview).

For the purposes of the thesis, we focus on workflows described at the level of their service instances, also known as concrete workflows (as opposed to abstract workflows).

Models of computation

Another major distinguishing factor between workflow languages is the model of computation adopted to orchestrate distributed resources. A model of computation (MoC) is a formal abstraction of execution in a computer.

1) A single model of computation. Most workflow environments fix the model of computation available to an e-scientist. Different experiments are modelled more cleanly with different MoCs because of their relative expressiveness and efficiency. Different uses of MoCs for scientific workflows include data-flow for pipeline compositions, e.g. gene annotation pipelines; continuous-time ordinary differential equation solvers, e.g. for Lattice-Boltzmann simulations in fluid dynamics; and finite state machines for modelling sequential control logic, e.g. in clinical protocols or instrument interaction. Many scientific workflow systems have adopted a *data-flow paradigm*, corresponding to variants of functional programming languages [YB05] [GL05]. For example, the MoC underlying the Taverna system's Scufl language corresponds to a

lambda calculus with a list monad [Tur06]. The data-flow paradigm, however, is not the only possible style of modelling workflows. Several projects rely on the semantics of the BPEL language, for instance the CaGrid, AstroGrid or CancerGrid projects.[10] *In this thesis we assume the use of the data-flow paradigm.* We relax this restriction in Chapter 7 to consider the use of multiple models of computation.

2) Multiple models of computation. The above workflow systems leave little flexibility to change model of computation as an experiment evolves. There are scenarios where a combination of MoCs is useful, e.g. a mixture of a time dependent differential equation model with dataflow. Most environments do not support experiments that mix multiple MoCs. A notable exception is the Kepler system [LAB+05] which allows for hierarchical composition of heterogeneous models of composition. Similarly, it is not known whether and how different workflow systems, adhering to different MoCs, can be used together. This interferes with the vision for intra- and inter-disciplinary collaboration in e-science.

For example, in genomic biology, gene annotation pipelines provide useful input to systems biology simulation models. Candidates for drug development found in cheminformatics simulations are plugged into bioinformatics annotation pipelines to retrieve the candidates' hazardous interactions within cells. The inability to mix MoCs also makes it more difficult to mix software workflows with physical systems such as sensor networks and electron microscopes, which have continuous dynamics. Moreover, mixing specialized MoCs for visualization (e.g. for animation) with, for example, time-based simulation, makes for more efficient execution and for better models. In addition, if we can mix MoCs, then we can introduce computational steerage of workflows. Representative use cases include: (i) selective extraction and analysis of proteins from public databases, combining finite state machines and dataflow and (ii) dynamically adapting model control parameters of Lattice-Boltzmann simulations in fluid dynamics by combining finite state machines and continuous-time ordinary differential equation solvers.

In such scenarios, using an integrated environment that supports mixing MoCs enables integrated provenance collection. In the fluid dynamics example, the provenance includes dynamic changes in the overall model as well as parameter sweeps within each model, covering the full range and variability.

We assume the existence of multiple models of computation in a workflow in Chapter 7.

[10] A similar dichotomy is also present in business workflow systems; see the recent survey by [MHH].

2.1.3 Formal definition

Summarising the assumptions we outlined above, this thesis only considers scientific workflows with the following properties:

Data flow oriented Given the target domain, the representation makes data an important element in the definition. There is no notion of time and no notion of events.

Workflow design We focus on concrete workflows, where it is known at design time which services will be used.

Semantics are optional The workflow definition may contain semantic (machine processable) descriptions. Details on the contents and different formalisms for these descriptions are given within.

Homogeneous execution semantics We focus on data flow workflows only and do not mix in other formalisms in a workflow definition. We relax this criterion in Chapter 7.

To simplify our discussion, we also assume the following:

Web services only We assume all services contained within the workflow can be exposed as a Web service. Workflows typically orchestrate Web services in conjunction with other kinds of services: other workflows, local components, different types of legacy distributed services (Taverna for instance accesses eight different types [LAWG05]); even humans are modelled as part of the process. Typically, a type of component is invented that generalises across all service types. For example, in the Taverna 2 system[11] and the Inforsense system[12], these components are called Activities; in Kepler they are Actors.[13] We assume that these components can be cast in terms of a Web service operation, based on the Web Service Definition Language WSDL, the standard for the description of a Web Service interface.[14] This would be the wrong assumption to make when other service paradigms are in use that cannot be cast elegantly in WSDL, e.g. REST-style components.

[11] Web site: `taverna.sf.net`
[12] Web site: `www.inforsense.com`
[13] Web site: `www.kepler-project.org`
[14] Web site: `www.w3c.org/2002/ws/desc`

Opaque nested workflows We assume that any sub-workflows present in a workflow are represented as a regular service where no knowledge is available about the internal structure of that service.

With the above restrictions in mind, this section formally defines what we mean by a workflow. We define a workflow w as the following triple:

$$W = \langle nameW, OP, DL \rangle$$

where *nameW* is a unique identifier for the workflow, *OP* is the set of operations from which the workflow is composed, and *DL* is the set of data links connecting the operations in *OP*. In what follows we discuss the contents of the following *workflow elements* in detail:

- *operations* which have *input and output parameter types*;
- *data links*;
- *overall input and output parameter types* and
- *workflow fragments*.

Operation An operation $op \in OP$ is defined as:

$$op = \langle nameOp, loc, in, out \rangle$$

where *nameOP* is the unique identifier for the operation, *loc* is the URL of the Web service that implements the operation, and *in* and *out* are two sets representing the input and output parameter types of the operation, respectively. In our description of an operation, there is no description of the relationship between inputs and outputs.

Parameter types A parameter type provides information on the structural type and semantic type of a given operation input type (belonging to the set *op.in*) or a given operation output type (belonging to *op.out*). A parameter type is defined by the couple:

$$\langle op, p \rangle$$

where op is the operation to which p belongs, and p being the triple:

$$p = \langle nameP, str\_type, sem\_type \rangle$$

with *nameP* as the parameter type's identifier (unique within the operation) and *str_type* as the parameter type's structural data type. We assume an XML type system, so that parameter data types may be either simple types, such as *xs:string* and *xs:int*, or complex types, built from simple ones. *sem_type* specifies the semantics of the parameter type. While the structural data type of a parameter type can be retrieved from the WSDL document describing the Web services, its semantic type corresponds to an ontology concept that is chosen when annotating the semantics of the Web service. We do not assume that the semantic types contain information about pre- and postconditions. We also do not assume they contain knowledge of the relationship between inputs and outputs.

Data links Let *IN* = $\cup_{(op \in OP)}$*op.in* be the set of input types of all the operations that constitute a workflow, and *OUT* = $\cup_{(op \in OP)}$*op.out* be the set of output types of all its operations. The set of data links connecting the workflow operations must then satisfy:

$$DL \subseteq (OP \times OUT) \times (OP \times IN)$$

A data link relating the output type *o* of the operation *op1* to the input type *i* of the operation *op2* is therefore denoted by the quadruple *(op1,o,op2,i)*. If *(op1,o,op2,i)* belongs to DL then *(op1, o)* is an output of *op1* and *(op2, i)* is an input of *op2*.

Figure 2.3 shows the example data flows *W1* and *W2* visualised as Taverna workflows. The set of operations for W1 corresponds with W1.OP = $\{op1, op2, op3\}$ and for W2 with W2.OP = $\{oq1, oq2, oq3\}$. The datalinks are defined for the first workflow by $W1.DL = \{< op1, o1, op2, i2 >, < op1, o1, op3, i3 >\}$. For the second workflow, the set equals $W2.DL = \{< oq1, o1, oq3, i2 >, < oq1, o2, oq3, i3 >, < oq2, o4, oq3, i2 >$ $\}$. Note that the diagram only depicts identifier names – no information is included with respect to an operation's location or the parameters' syntactic and semantic types (for brevity's sake we do not include these in the text either).

Overall input and output types For any given workflow, those input types to operations which are not connected to any data link are considered to be the input types to the workflow as a whole. Similarly, the set of operation output types which are not connected to any data link are considered to be the output types from the whole workflow. Given a workflow *W*, we use the terms *W.in* and *W.out* to respectively denote the input parameter types and the output parameter types of *W*. In the example of Fig. 2.3,

$W1.in = \{< op1, i1 >, < op2, i3 >\}$,

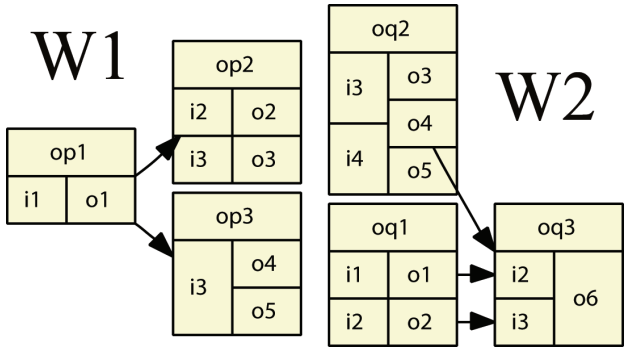


Figure 2.3: Data flow workflow examples based on a Taverna layout.

$W1.out = \{< op2, o2 >, < op2, o3 >, < op3, o4 >, < op3, o5 >\}$,
$W2.in = \{< oq1, i1 >, < oq1, i2 >, < oq2, i3 >, < oq2, i4 >\}$ and
$W2.out = \{< oq2, o3 >, < oq2, o5 >, < oq3, o6 >\}$.

For simplicity, henceforth we shall write "parameter" as shorthand for "parameter type," and similarly we will speak of "inputs" and "outputs" when discussing "input types" and "output types."

Workflow fragments Workflows have workflow fragments. To define these, we introduce an auxiliary function $ContainsW$, defined as:

$ContainsW$: *(W × W)* $\longrightarrow$ *Boolean*

where

$$ContainsW(W_1, W_2) = \begin{cases} true & W1.OP \subseteq W2.OP \wedge W1.DL \subseteq W2.DL \\ false & \text{otherwise.} \end{cases}$$

We define a workflow *F* to be a workflow fragment of workflow *W* when the set of operations and data links of workflow *W* subsumes those of workflow *F*. In other words, when

$$ContainsW(F, W) = true.$$

When the condition of shared datalinks (i.e., the same ordering between services) is dropped, we obtain a weaker notion of what a fragment is. $ContainsOps$ is defined as:

$ContainsOps$: *(W × W)* $\longrightarrow$ *Boolean*

where

$$ContainsOps(W_1, W_2) = \begin{cases} true & W1.OP \subseteq W2.OP \\ false & \text{otherwise.} \end{cases}$$

2.2 Workflow re-use and repurposing

A key challenge for scientific workflows lies in supporting the rapid assembly of workflows from disparate services and their *re-use* in various scenarios. Service re-use is a desirable goal of Service Oriented Architectures and Web services middleware. Workflow re-use has received less attention, yet has the potential to: i) reduce workflow authoring time by less re-inventing of the wheel; ii) improve workflow quality through re-use of established and validated workflows rather than re-invention of new, and potentially error-prone, ones.

2.2.1 Definition

We distinguish between workflow *re-use* and the *repurposing* of workflows.

- A user will **re-use** a workflow or workflow fragment that fits their purpose and could be customised with different parameter settings or data inputs to solve their particular scientific problem.
- A user will **repurpose** a workflow or workflow fragment by finding one that is *close enough* to be the basis of a new workflow for a different purpose and making small changes to its *structure* to fit it to its new purpose.

Note that this definition implies that re-use at one level of workflow abstraction may lead to repurposing at another level of workflow abstraction. For instance, the re-use of an abstract workflow by changing a parameter may lead to structural changes in the concrete workflow associated with that workflow (and thus implies its repurposing). A practical example is the re-parametrisation of an abstract workflow that performs pair-wise sequence similarity to do multiple sequence similarity instead. This might be realised at the level of the concrete workflow by replacing a BLAST Web service with a ClustalW Web service.

Figure 2.4 provides an example of repurposing based on two dataflows. It shows the insertion of service **c** from Workflow 2 in between the previously connected services **a** and **b** of Workflow 1. In terms of the underlying bioinformatics, Workflow 1

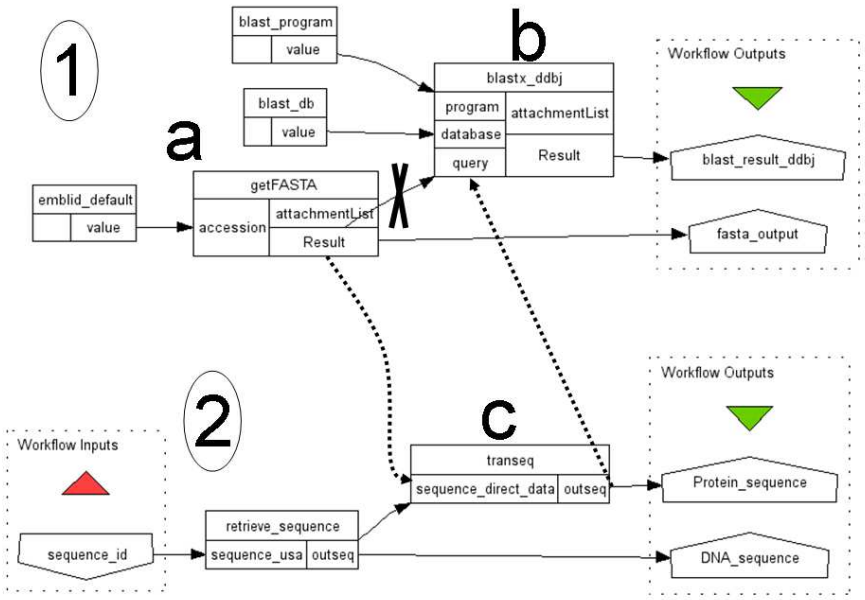


Figure 2.4: Example of an insertion based on two Taverna workflows

is extended with the Transeq service, which changes the workflow from a pipeline for measuring similarity of DNA sequences into one that analyses similarity of peptide sequences.

Observe how Workflow 1 provides useful input to locate Workflow 2 in a repository: one can concentrate the search on those service compositions that acccept service **a**'s output and produce service **b**'s input. Finding compatible insertions is one type of discovery that supports the repurposing of dataflows. The other types are the discovery of replacements and the discovery of extensions that append or prepend a workflow.

While re-use is intuitively well understood, repurposing is less so. It inherently requires metrics for measuring similarity, and repurposing actions to rework the workflow by adding, removing, or replacing steps, or by altering the control structure. Repurposing may involve techniques to provide a user with suggestions as to what are the relevant pieces of workflow for their experiment, like "Based on the services and structure of your workflow, it looks like you are building a gene annotation pipeline. Other users have found this collection of fragments useful for that."

2.2.2 Case studies in workflow re-use

To illustrate the concepts, we present case studies of workflow re-use drawn from different scientific disciplines. Based on these we formulate necessary requirements for workflow re-use.

We start with several case studies from the bioinformatics domain. Cases from other disciplines are also presented, with less detail. The latter stem from the pharmaceutical industry, earth sciences, geology, fluid dynamics and chemistry.[15]

To obtain the data, we collaborated with biologists, bioinformaticians and developers that use the ^{my}Grid project's Taverna workbench. We also selected a cross-section of middleware projects from the e-Science programme in the United Kingdom, which was the first of its kind [HT02]. Further, we interviewed core developers from InforSense (a spin-off from the DiscoveryNet e-Science project [SEG+03]), Geodise [TCS+04], Triana [MWG04], the Sedna project and people from the USA-based Kepler project [LAB+05].

Bioinformatics

Taverna users have built in excess of 500 workflows to support them in doing bioinformatics. Through our interactions with the community, (i) we identified practical cases of workflow re-use and (ii) we collected user experiences of re-use.

1) Workflow re-use cases. We have worked closely with members of three bioinformatics groups to observe workflow re-use. They actively research the following topics:

1. Investigation of Williams-Beuren Syndrome [STW+04]. Members of St Mary's Hospital Academic Unit of Medical Genetics, at the University of Manchester, have developed workflows (i) to identify any newly deposited and relevant genome sequences in public sequence databases (ii) to characterise any genes in those new sequences using analysis tools (iii) to gather related information from other databases (iv) likewise to characterise proteins that will be produced from those genes.
2. Investigation of the genetic basis of Graves' Disease [LHJ+04]. Members of Institute of Human Genetics at the University of Newcastle have developed a set of workflows to statistically analyse data showing the changed expression of genes in affected thyroid tissue, followed by characterisation of those genes.

[15] The survey form is available from `www.myexperiment.org/benchmarks`

3. Investigation of Trypanosomiasis (sleeping sickness) resistance in cattle [FHW+07]. Members of the Bio Health Informatics Group at the University of Manchester built workflows which identified a metabolic pathway for which its correlating gene is believed to play a role in resistance to Trypanosomiasis. Manual analysis on microarray and Quantitative Trait Locus (QTL) data previously failed to identify this gene as a candidate.

Figure 2.5 shows the witnessed re-use of workflow fragments. Fragments have been re-used between researchers that are active within the same research project. Re-use also occurred between groups from separate projects. In all cases, the workflows were discovered by word of mouth.

- In the case of the Graves workflow, the workflow took more than a year to create. During the process of building it, 56 workflows were created, most of which are overlapping versions and *re-used in* one shape or other in the *other versions*. The largest of these workflows contains 45 services.
- The research group who produced the Williams' syndrome workflow [STW+04] have seen a *dramatic drop in workflow authoring time* through the ability to repurpose workflow fragments from the Newcastle group.
- The figure shows the *re-use of fragments between research groups* active in the same project (from Graves to Williams), as well as re-use *between affiliated research projects* (from Williams to Trypanosomiasis). The Williams bioinformaticians were keen to extend their workflow with a protein annotation pipeline, as well as to introduce microarray analysis functionality. In turn, the Williams workflow itself became the subject of re-use for the Trypanosomiasis workflow, in particular the microarray analysis and gene prediction fragments shown on the figure. In case of the microarray fragment, in effect one sees the emergence of workflow fragment *propagation*. New re-use of fragments from the Williams workflow is planned to support research on the Aspergillus fungus.
- The workflow built to investigate Trypanosomiasis in cattle was *re-used without change* to run over a new dataset: Trichuris muris - the mouse whipworm. This identified the sex-dependent biological pathways involved to expel a parasite in mouse, which a two-year manual study previously was unable to verify.
- The arrows in the figure do not do justice to the difficulty it took to re-use the various fragments. Discovery of fragment functionality happened by word of mouth, and comparing and integrating fragments took extensive discussions between the workflow authors. *Repurposing* the workflow to investigate a different

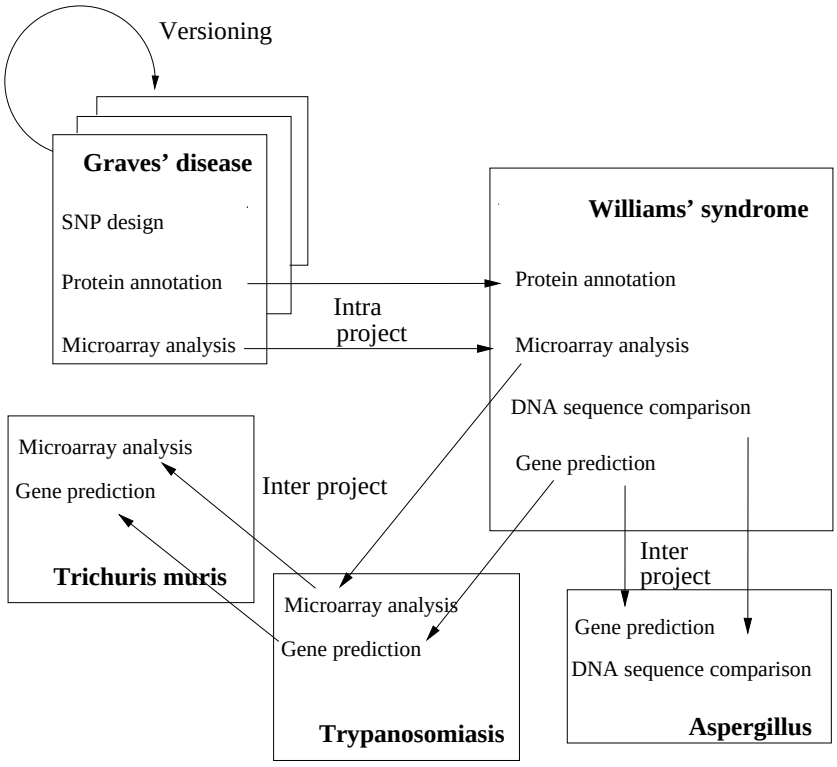


Figure 2.5: Different types of workflow re-use illustrated by a scenario from bioinformatics.

species meant the structure had to be adapted: certain services had to be replaced (for example, some gene prediction services are species-specific), others removed and still others added.

2) Workflow re-use experiences. We also queried a bigger sample of the bioinformatics academic community. In a survey with 24 bioinformaticians from 19 research laboratories (see Appendix A for the list of participants), we found that out of the 19 participants who indicated having built workflows before, 15 have re-used workflows. Of those 15 participants, 7 re-used workflows from third parties, 4 from a fellow research group member, 4 from a project collaborator and 2 from a colleague at the institute.

The respondents were polled about their *practical experiences during re-use*:

- All respondents believed that in most cases there is *not enough documentation* to understand a workflow.
- Ninety percent of respondents believed there are *no effective search tools to find relevant workflows*.

- For three quarters of respondents, some of the *services* in a workflow were (always or at least often) *non re-usable* due to the service being local to the original author. The same sentiment existed with respect to *services being down*.
- The majority of respondents believed there is *no way of trusting the analysis* performed by a workflow.
- Little under half of the respondents believed that often there are *not enough workflows* around, so they do not look for workflows.

All survey participants were asked about their *reservations for sharing their workflows for re-use by others*. They expressed the following main concerns:

1. Receiving proper *acknowledgements* for the work (36.8%) and
2. The workflow doing the job, but not being a piece of software they are very proud of (31.6%).

The following factors were deemed less important:

1. Being scooped (i.e. beat to obtaining results) by their own doing (15.8%);
2. Sharing the data that is obtained from the workflows (15.8%);
3. Sharing the data that feeds into the workflows (10.5%);
4. The brittleness of shared workflows, either due to the use of non re-usable local services or due to the volatility of remote services (10.5%);
5. Being able to share the workflow without others being able to establish how it works exactly (5.3%).

Other than bioinformatics, we also found examples of re-use in other domains.

Pharmaceuticals

Clients of InforSense, a commercial enterprise, have been building scientific workflows for several years. They exchange and extend workflows based on corporate intranet servers and e-mail lists. Given that these workflows are based on proprietary technology and often contain trade secrets, sharing with external parties has been limited.

Earth sciences

In Triana, the GEO power spectrum, a small composition of Java classes aimed at the direct detection of gravitational waves, has been shared between different research groups in the same department at Cardiff University.

Geology

The Kepler project so far has around 30 users which have built 10 workflows from a registry of 20 services. They have seen the redeployment of GRASS services for geospatial data management developed in one project (SEEK) to form a new pipeline for another project (GEON). This redeployment required a slight adaptation of the control flow.

Fluid dynamics

Geodise relies on the Matlab software environment for the orchestration of local Matlab functions which wrap distributed Grid resources. It offers access to some 150 functions, based on which 10 workflows were built to date. It re-uses both configurations and assemblies of Matlab functions (*i.e.* scripts) described by various authors.

Chemistry

The Sedna project at University College London has built a compute intensive workflow for chemistry, generating up to 1200 service instances concurrently. No re-use of this workflow has occurred. Sedna is the only project in our sample to use BPEL.

In summary, we observed that re-use of workflows and fragments of workflows is already happening. It is clear that re-use becomes harder as the conceptual and physical distance between parties increases. If re-use is to happen on a wide scale, a large set of workflows where people can draw from is key. In addition, detailed documentation and ways to search and *compare* the documentation of different workflows are needed. All of the above middleware projects offer a search mechanism to look for available services; *none* however *allow for the possibility to compare workflows descriptions.* In the next section we provide a systematic list of requirements for workflow re-use, drawing from the above use cases.

2.2.3 Workflow re-use requirements

The previous section highlighted several cases of workflow re-use in various scientific domains. They lead us to claim that workflow re-use in science only happens when the following three categories of requirements are met:

1. A community open to workflow re-use.
2. The availability of re-usable workflows.

3. Efficient and effective workflow discovery.

A community open to workflow re-use

Whether the culture in a particular scientific community is beneficial or detrimental to workflow re-use depends on three factors: (i) sharing attitude, (ii) re-use skill set and (iii) re-use motivation. Different communities are marked by different dynamics.

1) Sharing attitude. Sharing attitude is driven by the need for acknowledgments, the interaction with peers and a sense for quality assurance.

Scientists invest a lot of time in building workflows and are often hesitant to release workflows without *receiving proper acknowledgements* for their work. In a commercial setting this translates into a demand for formal Intellectual Property Rights agreements. Science has dealt with this problem before in the context of sharing experimental data. Scientists can publish in a journal when they release their data in public databases, with the inclusion of metadata. The submitted data is then anonymised to the extent that it is of no use for the direct competition or an embargo is imposed over the data, to ensure the original authors enough time to exploit the data. Authors of *in silico* experiments might publish their workflows in the same way. Further, there is the recent emergence of new license forms to publish scholarly work free to the public, such as the Science Commons initiative.[16] These may also prove appropriate as a scheme to assign scientists' contributions to workflows.

Interaction with peers also influences sharing motivation. Open source development of software invites contributions from other developers and is known to lead to virtuous network effects. In a similar vein, void of other incentives, the ratio between producers ("seeders") and consumers ("leechers") of workflows in a particular community will influence the motivation of producers to contribute. In an academic setting, producers often are consumers of others' workflows too, which stimulates a positive attitude towards sharing.

Given the importance of reputation in science, scientists also worry about *quality assurance* when releasing a workflow to a wider audience. Mechanisms to indicate and improve workflow reliability can reduce such fears.

2) Re-use skill set. "Re-use ability" or the IT skill set a scientist possesses determines the type of re-use she (or the team she may operate in) can technically manage. In this respect, we reiterate the distinction between *re-use*, where workflows and workflow fragments created by one user might be used as is, and the more sophisticated

[16] Web site: www.sciencecommons.org

repurposing, where they are used as a starting point by others.

3) Re-use motivation. The motivation of a scientist to consider workflow re-use as a solution as opposed to creating something from scratch is determined by her *skill set* (discussed above) and *the perceived quantity, quality and relevance of the workflow pool*. Whether perception matches reality depends on the mechanisms in place in a workflow publish and share infrastructure to communicate the quantity, quality and relevance of the workflow pool.

The quantity of available workflows signals to a re-user that there is a fair chance a relevant workflow will be available. *Workflow quality* signals that the re-use effort involved may be limited. Quality is measured in many ways, e.g. trust in the author, transparency of the performed analysis or the popularity of a workflow with peers. *Workflow relevance* is subject to the area a scientist works on. Some workflows are run only once, e.g. to compute Higgs boson in particle physics and become irrelevant as soon as the workflow has been run. Others remain relevant, e.g. bioinformatics pipelines monitoring updates to public sequence databases. Workflow relevance also relates to the technical re-usability of a workflow (discussed below).

The Availability of Re-usable Workflows

Availability of workflows is enabled by the provision of a publish and share infrastructure. Whether a shared workflow published on such an infrastructure is actually re-usable depends on (i) the availability and flexibility of the distributed autonomous services it orchestrates, (ii) the flexibility of its workflow language and the adopted model of computation and (iii) its reliance on data models.

1) A publish and share infrastructure. The workflow re-use case studies revealed that re-use occurs in many circumstances. Three categories of workflow re-use surfaced. The categories are based on the person doing the re-use: a workflow author re-using her own workflows (personal re-use or versioning), re-use by collaborators and re-use by third parties who the workflow author never met.

Personal re-use Building large workflows can be a lengthy process, sometimes taking years of time. This results in different versions of workflow specifications that co-exist in one location. Manually keeping track of the relationships is a challenging task, so versioning support is required. Versioning can be seen as a case of "personal re-use".

Re-use by collaborators Scientists are typically part of a research group and various

research projects, inside of which they exchange knowledge.

Re-use by third parties Third-party reuse is the kind of reuse envisaged by the e-Science vision for intra- and inter-disciplinary scientific collaboration.

This places additional requirements on scientific workflow system infrastructure, to:

- *Identify* re-usable services and workflows.
- Support the *generation* of re-usable services and workflows.
- *Register* and *advertise* available services and workflows in a community accessible location.
- *Annotate* these registrations.
- *Search* over service and workflow information by consumers.
- Effectively *re-use* discovered services and workflows.
- *Track* a service or workflow's re-use history.

The registration and discovery of workflows may transcend any specific workflow environment. This idea is being pursued in the development of the ^{my}Experiment.org infrastructure.[17]

In Figure 2.6, we show an extended scientific workflow life-cycle in the context of bioinformatics. For other scientific disciplines, the word "Bioinformaticians" can be replaced with "Workflow developers" and "Biologists" with "Workflow users."

1. Before embarking on a new design the author should consult a catalogue or *registry* of previously published workflows. Search facilities identify any existing workflows that perform exactly what they want, and is parameterised and instantiated as such; exactly what they want if it were re-parameterised; or is similar to their needs with slight modification. Once found it must be easy to transfer this workflow into a workbench for further editing and execution.
2. Workflows or their fragments are potentially edited; services are parameterised or bound to end points but rarely altered. Other services, workflows or workflow fragments are sought, or new ones are created. These too must be easy to integrate into the workflow design, and assembled, instantiated and executed within the workbench.
3. We cycle through this process until the scientist is happy, and the workflow has proven its worth.

[17] Web site: `www.myexperiment.org`

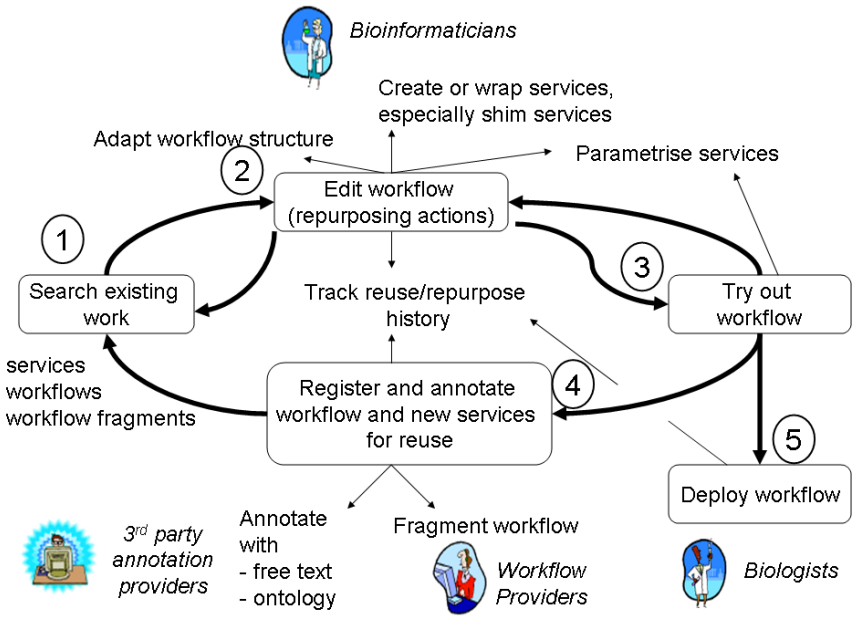


Figure 2.6: A scientific workflow life-cycle which extends beyond design and execution of workflows, to encompass discovery of existing resources for inclusion, and publication of its design.

4. It must then be a simple task to publish the workflow template to a registry, annotate with a description and additional knowledge on the suitability of the original workflow for this task, so that others can benefit. Conscientious users might partition the workflow into coherent fragments and publish those; otherwise an automated process might attempt the same. It must also be possible to go back and annotate the original workflow with this experience.
5. The user also publishes the workflow to a portal so that it can be run by scientists with no workflow expertise.

This life-cycle is dependent on descriptions of re-usable parts of a workflow. Which parts are technically re-usable depends on the distributed constituent services of a workflow, the workflow language and the reliance on data models. We discuss their impact in the following.

2) Distributed services. Restrictions on the *availability of services* (as a workflow's building blocks) creates a bottleneck for workflow creation and availability. First, domain users have strong opinions about the particular services that they wish to use. For them to be willing to create workflows, they need to have access to their favorite tools and databases from within the workflow environment. If these are not available

as services *accessible within the workflow environment*, they will use other technologies. All workflow projects introduced above (bar Sedna, which relies on the BPEL language which permits only Web services) offer access to types of services which are other than Web services. Second, service availability is also hampered by issues of *authentication, authorisation, accounting* and *licensing*. Third, the incorporation of *local services in a workflow*, be it as local components or Web services deployed behind a firewall, render a service unavailable for third parties. Repurposed workflows will need to replace those local services, unless they are either (i) Web-enabled upon publishing, (ii) made available for download in a public repository, or (iii) their functionality is made part of the workflow specification.

Services on the Web typically are outside the control of a workflow developer. The presented *service interface defines the limit* to which one can re-use the service: if the service interface does not support particular functionality, even though the underlying implementation of the service may, it is out of a developer's reach. This is a standard problem in object-oriented programming, where the solution has been to design objects with re-use in mind by providing rich interfaces.

3) Workflow language and model of computation. Workflow specifications can be hard to re-use, depending on the available support for *workflow evolution and adaptation* in the language. Workflows change as a result of (i) continuous process improvement, (ii) adaptations to changes in the workflow's environment, and (iii) customisation of a workflow to the needs of a specific case [JH98]. The workflow evolution literature typically considers (i) and (ii), and where (iii) is studied this is done from the perspective of a single organisation, and does not consider unpredictable reuse by third parties.

The saying "The nice thing about standards is that you can choose" also holds for scientific workflow languages. Each of the projects in the aforementioned survey uses its own language for orchestrating resources. This diversity reflects the different demands of the application areas and computer skills of users. When it comes to re-use and repurposing, it is desirable to have access to as wide a pool of workflows as possible. Hence, *workflow interoperability* becomes an issue. Research to identify, categorise and formally describe the different models of computation of a workflow is ongoing and limited. Multiple patterns for control flow, data, resources and exception handling have been identified [vdAtHKB03] and are only just being formalised [RtHvdAM06]. To our knowledge, this work has not been applied to compare and inform interoperation between scientific workflow systems. Also, it is unclear how

fragmented patterns join up to form entire models of computation. In section 2.1.2 we provided use cases for combinations of models of computation. Workflow developers would need to know how such models *compare* and how they can be *combined*. We start to address the issue in Chapter 7.

4) Data models. Workflows that are written against a data model can be more difficult to re-use. For example, the tight coupling between a workflow and a database requires re-users without access to this database to replicate the same conditions, which may be expensive. Rather than the information captured within the workflow diagram, knowledge of the data model becomes essential to understand what a workflow does. Further, reworking a data model to fit a new purpose is often non trivial. On the other hand, understanding the outputs generated by a workflow which does not use a data model can be a challenge, given that little integration between the outputs will be available. There is *a trade-off between data model repurposing complexity and workflow repurposing complexity*.

Effective and efficient workflow discovery

Determining the relevant workflows for a given re-use problem is challenging. Before we can specify requirements for workflow discovery, we explore the notion of workflow discovery in science in more detail.

2.3 Related work

Software re-use and the associated problem of finding relevant components have been a theme in software engineering for a long time [Kru92]. We survey how the scientific and business communities have embraced workflow re-use in particular.

Workflow technology originated in the business community to support office automation [GHS95]. We provide an overview of the adoption of workflow re-use in this community. We then switch communities and consider the situation in science.

2.3.1 Workflow re-use in business

Parallels between scientific workflow development and business workflow development have been drawn in [GWS+02]. There are also parallels to be drawn between how both communities re-use workflows. Just like scientific workflows, business-oriented workflows also contain parts that potentially are useful to others. Figure 2.7 (taken

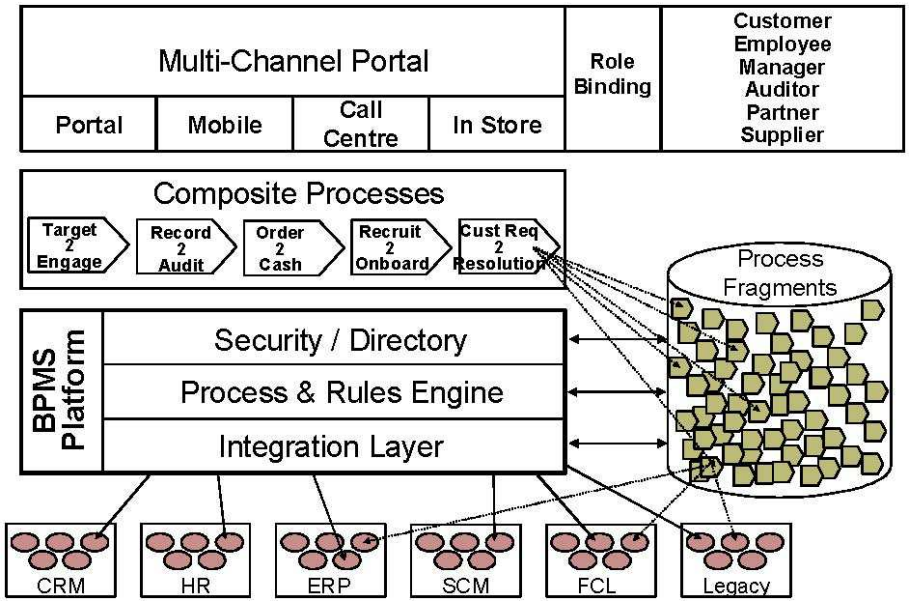


Figure 2.7: Business processes drawn from a pool of sub-processes.

from [MHH]) makes clear that business workflows are composed of re-usable fragments. Parallel to the types of scientific workflow re-use we observed in Chapter 2, in businesses there is *re-use of workflows within and across enterprises.*

1) Re-use within enterprises. Given commercial pressures and issues of Intellectual Property Rights, workflow re-use in business *traditionally* has been *restricted to the context of the enterprise*. For example, in the literature, Blin *et al.* [BMW03] present a vision and a highly parameterisable language for re-usable workflow building blocks, to be shared within the enterprise. In a recent survey of commercial workflow systems [MHH], the authors discuss how current day systems handle sub-workflows during workflow design and enactment. The context for this discussion is a single enterprise. The authors observe considerable variability in the way systems tackle process adaptability, based on whether vendors thought the problems were all about the control of employees or, at the other end of the scale, whether they sought to enable knowledge workers to achieve their goals. This is analogous to how the drivers for workflow re-use workflows differ between scientific communities.

Interestingly, the survey notes that "many vendors provide *templates* that give the business a start point in developing specific types of applications. For example, a vendor could provide *proprietary business rules for popular industry processes*, or even offer *software components* to facilitate the development of a specific type of business process. Some go so far as providing support for entire applications focused on a specific vertical or horizontal application."

2) Re-use across enterprises. In recent years, workflow systems for the commercial enterprise have started to consider *re-use of workflows across enterprises*. The above survey also found that some vendors provide *industry frameworks* that define vocabularies and metrics for specific types of processes, for example for the insurance industry or for companies working on supply chain systems. This feature chimes with the vision behind the RosettaNet initiative, which offers a library of Partner Interface Processes (PIPs) available for re-use.[18] PIPs specify templates of *process interactions between companies* active within a certain industry.

A different kind of workflow re-use across enterprises is enabled by the MIT Process handbook [MCH03]. As a contribution to the organisational studies literature, it has documented some 5000 real world business processes (which cannot be enacted computationally). The goal is to support *innovation in organisations* by classifying known business cases.[19]

2.3.2 Workflow re-use in science

We concluded from our survey of re-use cases in Chapter 2 that the notion of workflow re-use is relatively new for the scientific workflow community. A prototype system for re-use of scientific workflows based on digital libraries is described in [MPAD+05]. Few workflow repositories have been made publicly available. Examples of private workflow repositories are the Inforsense Customer Hub[20] and SciTegic's Scientific Applications.[21] The academic Kepler project [LAB+05] is building a platform with scientific workflow re-use in mind. It offers some 30 public workflows covering the earth sciences, ecology, chemistry and biology.[22] Work in this thesis has inspired and contributed to the establishment of the *my*Experiment.org portal for scientific workflows.[23] The site contains over 300 freely available workflows, supporting experiments in biology, chemistry, the social sciences and music.

[18] Web site: `portal.rosettanet.org/cms/export/sites/default/RosettaNet/Downloads/RStandards`
[19] Available from: `ccs.mit.edu/ophi/index.htm`
[20] Web site: `hub.inforsense.com`
[21] Web site: `www.scitegic.com/products`
[22] Web site: `library.kepler-project.org/kepler`
[23] Web site: `myexperiment.org`

2.4 Summary

This chapter provided basic terminology and definitions for the central concepts of the thesis: scientific workflows, workflow re-use and workflow repurposing. The notion of workflow discovery is introduced in the next chapter.

Chapter 3

Workflow discovery

In the previous chapter we scoped our interest to the re-use of concrete scientific workflows (i.e. workflows that include information about the service instances they orchestrate). This chapter provides a *definition* of workflow discovery and clarifies *the link of workflow discovery with re-use and repurposing* and *the link between discovery and composition*. Workflow discovery potentially means many things. We discuss the assumed *information need* for it and what parts of a workflow are to be searched over. Knowledge of the information need allows us to specify a specific class of *workflow discovery tasks formally*. We formulate these in terms of workflow *matching types*. Finally, the definition of matching types enables us to organise *existing work* on workflow discovery techniques in a structured way.

3.1 Definition

We introduce the following definition of *workflow discovery*.

> Workflow discovery is the *process* of retrieving orchestrations of services to *satisfy user information need*.

Workflow discovery is a process Workflow discovery is a process that is manual or automated. Manual workflow discovery does not scale well for the individual faced with an increasing number of workflows, but its observation potentially reveals problem-solving patterns that are useful to automated techniques. Automated workflow discovery requires electronic input to enable the process, such as textual queries, navigation based on hyperlinks, tag clouds or even known examples of workflows. Effective automated support of the workflow discovery process requires an understanding

of its technical requirements. We investigate these in Section 3.2.

Satisfy user information need Our target users are *scientists* looking for existing workflows that support their research. To be able to satisfy them, we need to document their information need and to evaluate how well retrieval techniques fulfill it. Interpreting the definition in the context of science implies we need a better understanding of what scientist information need entails. We pursue this in Section 3.3.

In what comes next, to ground the concept of workflow discovery in relation to existing work, we clarify the conceptual relation between workflow discovery, re-use, repurposing and composition. We provide a detailed discussion of existing work at the end of the chapter.

3.1.1 Relation to workflow re-use and repurposing

The user information need of scientists has implications on the type of discoveries to be made. In the previous chapter we documented their information need based on several case studies where scientists recycled workflows created by others. We found it useful to draw a distinction between *workflow re-use*, where workflows and workflow fragments created by one scientist might be used as is, and the more sophisticated *workflow repurposing*, where they are used as a starting point by others.

It is important to realise that the difference between workflow re-use and repurposing leads to different requirements for the discovery step. Whereas re-use requires finding workflows that are similar to a given user query ("Find a workflow that produces protein sequence."), repurposing requires finding both similar workflows ("Find a workflow able to replace my faulty workflow fragment.") and complementary ones ("Find a workflow that extends my current annotation pipeline with a visualisation step.").

3.1.2 Relation between discovery and composition

The workflow re-use lifecycle presented on page 45 (including Figure 2.6) features the discovery of existing work as the first step in the cycle, followed by an editing step to make it fit for purpose.

To clarify the role of workflow discovery during workflow re-use, we need to be precise about the semantics of the word "discovery" and its relation with "composition." We rely on the following definition of these concepts.

Discovery is the process of finding, ranking and selecting *existing* artifacts. Discovery operates over descriptions of atomic or composite artifacts.

Composition is the process of combining artifacts into a *new* working assembly. It is performed either manually, semi-automatically or automatically. Composition typically combines discovery with integration. If either activity involves manual intervention from a human, composition becomes non-automated.

The reader will recall that on page 35 we defined what we mean by workflow re-use and workflow repurposing. Workflow re-use assumes only parameter changes are made to a found workflow. Workflow repurposing involves making changes to its structure, including deletions, extensions, replacements or insertions.

In terms of discovery and composition, workflow re-use requires only discovery (of existing workflows). In contrast, workflow repurposing involves a discovery step as well as, in most cases, an integration step and therefore should be seen as a form of composition.[1]

Work in this thesis focuses on supporting *semi-automated* workflow re-use and repurposing. Our choice to support only semi-automation draws on the observation made in [LBW+04] that scientists in general are reluctant to relinquish control over the construction of their experiments. We aim to support scientists' activities, not replace them. It is the difference between providing automated decision making and providing automated decision support.

The automated decision support in our case is to be found in the *discovery* step. This step should automatically generate clues as to what would be the best workflows or fragments for a human to consider based on the existing workflows. Automated discovery does not imply automated selection, however. For workflow *re-use*, we leave the selection of the most relevant workflow amidst the discovered ones up to the workflow developer. Similarly, for workflow *repurposing*, the selection and integration part is left up to the workflow developer. Hence a newly repurposed workflow is the result of semi-automatic composition. Workflow discovery techniques built with repurposing in mind in this sense are to be seen as *composition-oriented discovery* techniques and sit in between automated discovery and automated composition.

[1] The exception being the case where workflow repurposing entails only making deletions to a found workflow; in this case no integration between a workflow and a workflow fragment or a service occurs.

3.2 Workflow discovery requirements

If we are to support the specified workflow discovery tasks with automated techniques, a number of requirements need to be fulfilled. Being able to handle the available number of workflows, having the right kind of documentation available to assess workflow relevance and an effective ranking of potential solutions are key requirements for finding relevant workflows.

3.2.1 Scalable discovery techniques

As the workflow pool grows, human perusal of the available pool becomes impossible. Automated techniques that support users in retrieving relevant workflows will need to combine fast performance with accurate precision and recall. Chapter 6 explores the recall and precision of a range of discovery techniques. The thesis does not investigate the performance of techniques in the face of an ever growing pool of workflows.

3.2.2 A comprehensive discovery model

The focus of a workflow author typically is on the scientific task performed by component operations, the workflow as a whole, and the experience others have gained in using them. At least in initial design, authors are less concerned with technical information about how each operation is invoked. A structured description of this scientific functionality provides more scope for automated assistance. Designing models that can capture what is being done, why, and what has been tried before but failed, is a big challenge.

In Section 3.3, we provide a first overview of the information scientific users need to do workflow discovery. Through a series of user experiments, the next chapter will explore in detail how bioinformaticians go about discovery and which aspects of workflow documentation they deem important. This knowledge is very useful for the selection and design of workflow discovery techniques, which is the topic of Chapter 5.

3.2.3 The process knowledge acquisition bottleneck

The more comprehensive a discovery model, the more effort is involved to author the model. Scientists are reluctant to manually populate any model of an experiment. The challenge then is to get the most benefit out of descriptions already available in a workflow by default and to be knowledgeable about the benefits richer descriptions bring

to the table. Chapters 5 and 6 explore the impact of techniques that rely on different types of descriptions, in particular natural language and Semantic Web languages. A related question, left unexplored in this thesis, is how to reduce the cost of acquiring annotations by means of automated techniques.

3.2.4 Lack of workflow fragment rankings

Once workflows and annotations based on the workflow model are created, one can query these for relevant fragments. As workflow fragment discovery is about retrieving those fragments that are "close enough" to a user's context, the notion of rankings is inherently present. Fragment rankings are the result of applying a series of metrics to workflow annotations based on a query mechanism.

Challenges lie ahead in defining metrics for workflow similarity and workflow complementarity based on the workflow-specific conditions specified in Section 3.4 and in evaluating their suitability.

3.3 Information need for workflow discovery

Scientists search for scientific workflows with different information needs. We envisage they pursue the following three goals. They will inform the content of discovery models.

- Construction of an in silico analysis.
- Linking work done in vivo and in vitro with work done in silico.
- Validation and extension of publications that report on an in silico analysis.

Scientists work to achieve their goals from within different contexts and each of these contexts potentially contains helpful information for finding relevant workflows. For each of the above goals, we discuss examples of contexts which may contain useful sources of information to inform workflow discovery.

3.3.1 Construction of an in silico analysis

1) Web search engines. The construction of an in silico analysis by means of a workflow typically starts from an initial idea. Scientists can feed the idea to a (possibly specialised) Web search engine and filter the results for workflows. The Google search engine for example can be tuned to return only results from part of its overall index.

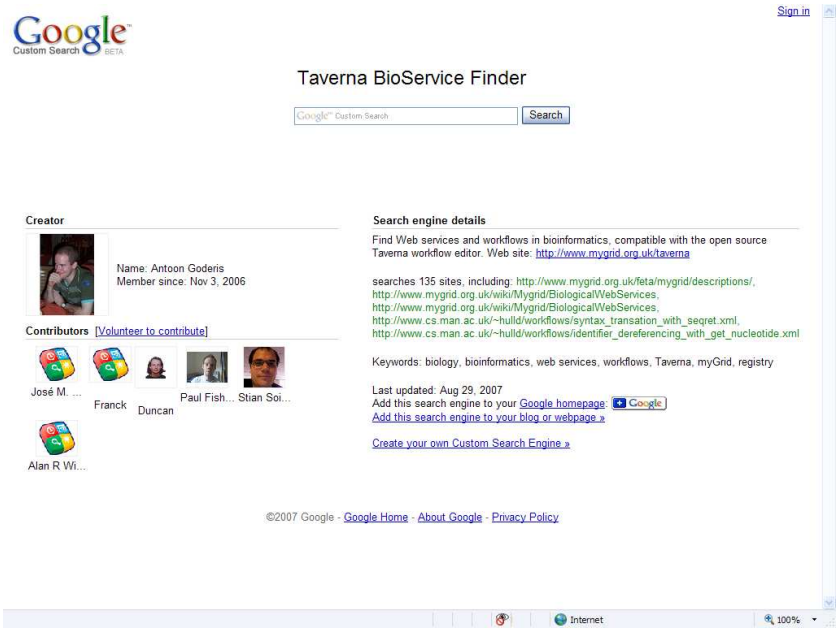


Figure 3.1: A customised version of the Google search engine for bioinformatics services and workflows.

Figure 3.1 shows a version that filters on workflows and Web services for bioinformatics.

2) Brainstorming tools. If the initial idea for a workflow happened to be captured during an electronic brainstorming session, this information can also be exploited to find related workflows. Figure 3.2 provides the hypothetical example of a "mind map" for a workflow investigating Trypasonomiasis.[2]

3) Workflow editors. If a scientist knows some of the data or services that will be

[2]The example was created with the Mindjet MindManager Pro software.

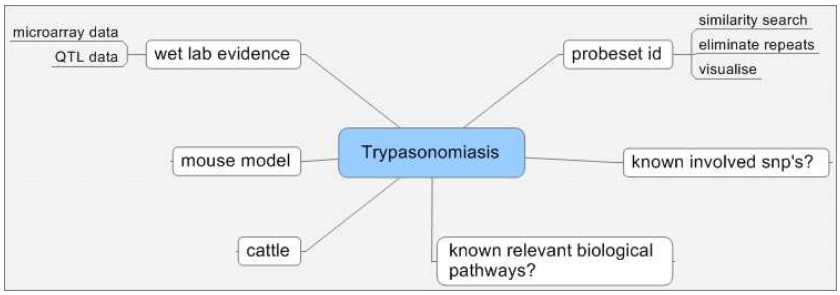


Figure 3.2: A "mind map" for a workflow investigating Trypasonomiasis.

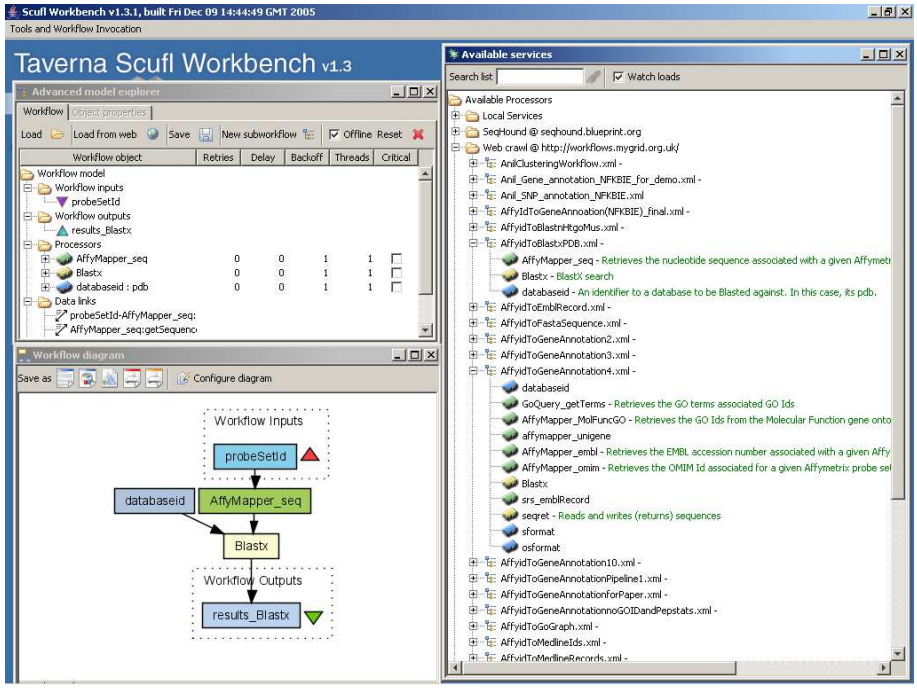


Figure 3.3: A query by example approach for workflows illustrated.

used in the analysis, she can construct an incomplete workflow based on this information and use the incomplete workflow as input to a "query by example" approach for finding workflows. In Figure 3.3 a small workflow created with the Taverna workbench is shown on the left. On the right, a collection of available workflows is listed. This collection could be filtered in terms of its relevance to the small workflow.

4) Sharing and collaboration Web sites. Sharing and collaboration sites for workflows should be the ideal place to find a workflow. In addition to the above mentioned sources of information to guide the search for workflows, the social networking angle these sites offer would enable scientists to find workflow authors with like-minded research interests. ^{my}Experiment.org (Figure 3.4) is an early example of such a site.

3.3.2 Linking work done in vivo and in vitro with work done in silico

The successful linking of scientific work done in the physical world with on-line analyses can validate hypotheses from the physical laboratory and generate new ones. For example, in biology, in vivo and in vitro work in the "wet" laboratory generates leads

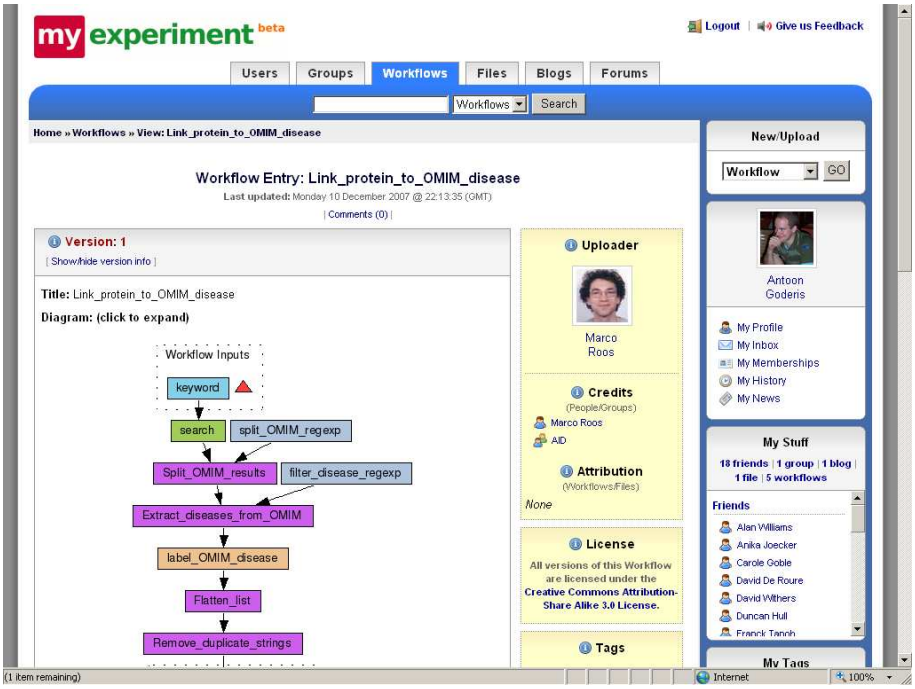


Figure 3.4: The ^{my}Experiment site for workflow sharing and collaboration.

that sometimes can be confirmed by running an in silico analysis on-line. For an example of this interaction see [FHW+07]. In vivo and in vitro protocols are increasingly published on collaboration Web sites such as OpenWetWare[3] (Figure 3.5) and discussed on specialised science blogs such as Nodal Point.[4] The protocols can serve as the basis for workflow searches.

3.3.3 Validation and extension of publications.

The submission of detailed experimental results alongside the submission of a scholarly publication facilitates the reproduction of the reported experiment. When publications report on the processing of on-line data, a workflow detailing the processing could be attached to the submission. This should ease the validation, reproduction and potential extension of the submitted work. The availability of open access journals at places like PubMedCentral[5] (Figure 3.6) or the Public Library of Sciences[6] eases establishing links between publications and their relevant in silico counterparts.

[3]Web site: `www.openwetware.org`
[4]Web site: `www.nodalpoint.org`
[5]Web site: `www.pubmedcentral.org`
[6]Web site: `www.plos.org`

Figure 3.5: OpenWetWare: sharing in vivo and in vitro protocols.

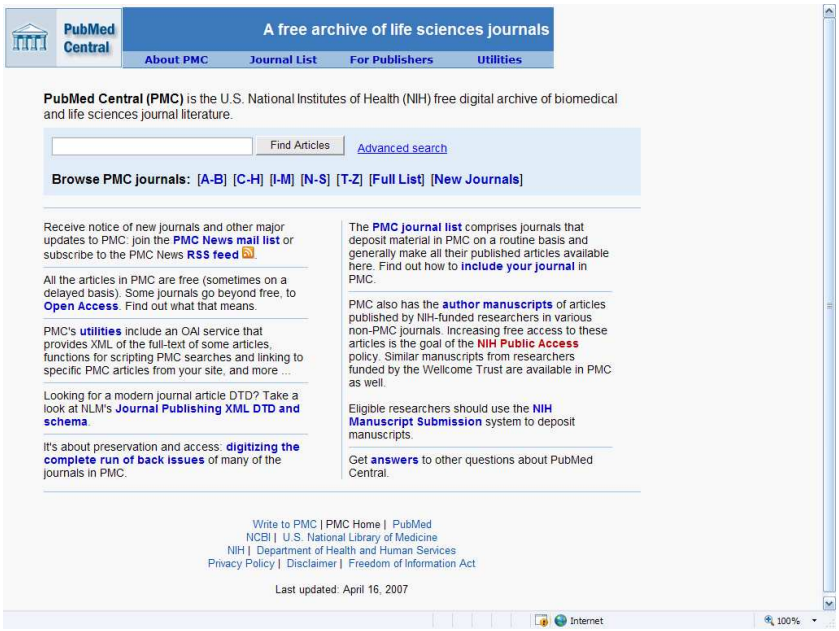


Figure 3.6: PubMedCentral: a portal for open access journals in biomedicine.

3.4 Workflow discovery matching types

Knowledge of the user context allows us to be more precise about the workflow discovery tasks we seek to support. We present the result of a survey with bioinformaticians about how they would perform discovery. We then zoom in on a particular class of workflow discovery tasks. The tasks focus on data flows and rely on the notion of workflow matching conditions. We specify different types of workflow matching.

3.4.1 Workflow discovery by signature and structure

We probed a sample of bioinformaticians about the level of detail they would wish to see while discovering workflows. During two ^{my}Grid Taverna User Day events (5-6 May and 15 November 2005), 21 out of a total of 45 participants completed a requirements analysis questionnaire, of which 15 were bioinformaticians and six software developers.[7]

1) Discovery based on workflow signature. When asked how they might look for workflows, four participants wrote they would search for workflows in the same way as they do for services. The question of workflow discovery then becomes one of discovery of a Web service based on its signature. We asked users how important various service search criteria were to them. All given criteria came out as relevant. The criteria, in order of decreasing relevance to participants, were: *Task, Input, Output, On-line documentation, Service Provider, Underlying Resource used* (*e.g.* a particular database), and *Algorithm used* (*e.g.* a particular clustering algorithm). For details we refer to the on-line survey data. In an optional *Other:* field, users could enter additional criteria. A few users entered Quality of Service parameters here, in particular performance and reliability measures.

2) Discovery based on workflow structure. When asked how they might look for workflows, five of the participants indicated they would not only rely on using a workflow's signature. They expected to be using structural information during search, such as the services contained in a workflow, the specific subtasks addressed by the workflow or to start from existing template workflows. This suggests a type of discovery based more on the shape or structure of a workflow, using more behavioral type of information.

[7]The survey data are available from the ^{my}Experiment Wiki at `www.myexperiment.org/benchmarks`.

We then asked users to rate the following criteria which also rely on structural information. The criteria are presented here in decreasing order of relevance as assigned by users.

- *Data flow* Given a set of data points, have these been connected up in an existing base of workflows? Data flow queries came out as very important.
- *Service flow* Given a set of services, have these been connected up in an existing base of workflows? Service flow queries also came out as very important.
- *Workflow similarity* The use of similarity to identify relevant workflows came out as important.
- *Use of specific control flow constructs* Queries based on specific control flow constructs, such as the appearance of looping and conditionals in a workflow, were considered important. This was against our expectations and indicates that users should be enabled to query specifically for such constructs.

3.4.2 Structural workflow matching types

In the thesis we shall focus on discovery based on workflow structure. We identify the workflow matching types involving workflow structure in this section and re-use them in the next section to formally specify workflow discovery tasks specifically involving structure.

In other words, we focus on that part of the workflow discovery problem that differs from the much better researched problem of Web service discovery - i.e. those aspects that involve structural properties of a workflow. We specifically investigate the possibilities for matching data flow workflows based on the workflow definition adopted in Section 2.1.3. Recall the example data workflows we discussed earlier. They are featured again in Fig. 3.7. It features workflow W_1 with $OP = \{op1, op2, op3\}$ and workflow W_2 with $OP = \{oq1, oq2, oq3\}$.

We distinguish between two workflow matching types:

Similarity based matching We wish to determine whether W_1 and W_2 are similar (or identical). Matching here will rely on metrics defined in terms of workflow element identity.

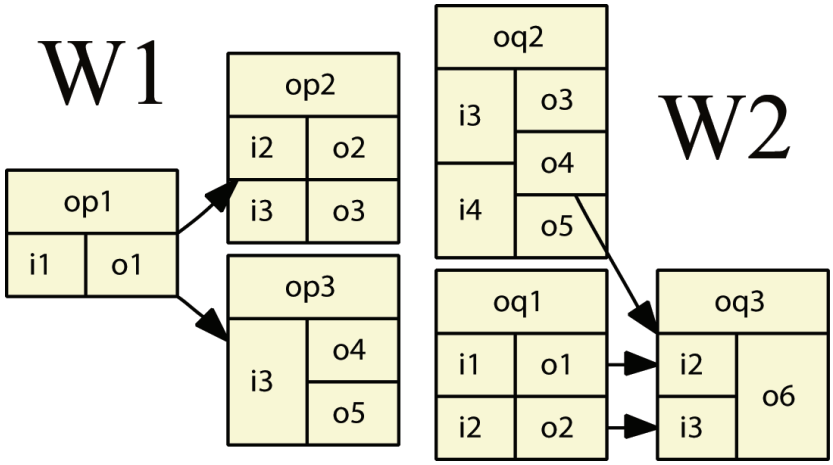


Figure 3.7: Matching workflows W_1 and W_2: which workflow elements to compare and how?

Complement based matching We wish to determine whether W_1 and W_2 are complementary. Matching in this case will rely on metrics defined in terms of workflow element complementarity.

We define both types in terms of how they compare the workflow elements introduced in Section 2.1.3. Between workflows W_1 and W_2 in Figure 3.7, a certain level of (mis-) match exists between the types of workflow elements: (i) matches between parameters, (ii) between operations, and (iii) between workflows.

In this section we develop *definitions for the matching types at each level*. This will bring out the dependencies between the levels. We start by defining matching types based on similarity at the parameter, operation and workflow level. We then introduce matching types based on complementarity at the same levels. Note that we are not developing algorithms to support discovery tasks. We merely define *basic conditions for similarity and complementarity of data flow based workflows*. We claim this should be useful to compare discovery techniques and to inspire designers of discovery techniques.

Section 3.5 will then construct *definitions of the discovery tasks in terms of the matching types*. The matching types will bring out points of commonality and difference between the discovery tasks.

3.4.3 Similarity-based matching

Workflows are described at the level of parameters, operations and the entire workflow. How these elements are to be compared to establish workflow similarity, either overall or of its parts, depends on the adopted view of similarity as a concept.

What is similarity?

Similarity can be approached in multiple ways. What are these approaches and which of them is the best one for the workflow context?

Approaches to similarity. Cognitive scientists distinguish four major ways of measuring similarity [Gol01]: featural, alignment-based, geometric and transformational. We do not discuss all approaches here but to at least illustrate the diversity we contrast the featural approach to similarity (using the Tversky measure [Tve77]) with the alignment-based approach.

(i) The featural approach. As Goldstone writes in [Gol01], "the Tversky measure determines similarity by matching features of compared entities, and integrating these features by the formula:

$$S(A, B) = \theta f(A \cap B) - \alpha f(A - B) - \beta f(B - A)$$

The similarity of A to B, $S(A, B)$ is expressed as a linear combination of the measure of the common and distinctive features. The term $(A \cap B)$ represents the features that items A and B have in common. $(A - B)$ represents the features that A has but B does not. $(B - A)$ represents the features that B, but not A, possesses. The terms θ, α, and β reflect the weights given to the common and distinctive components, and the function f is often simply assumed to be additive." In the case of workflows, features might be the operations used in a workflow, the shape of a workflow diagram, the authors of a workflow etc.

(ii) The alignment-based approach. Goldstone continues [Gol01]: "Featural models of similarity are not well suited for comparing things that are richly structured rather than just being a collection of coordinates or features. In such cases, comparing things involves not simply matching features, but determining which elements correspond to or align with one another. Matching features are aligned to the extent that they play similar roles within their entities." For example, suppose workflow 1 which has the overall output *3D structure of a protein* and workflow 2 which has the overall input *3D structure of a protein*. They both share the feature *3D structure of a protein*,

but this matching feature may not increase their similarity much because workflow 1's output role does not correspond to workflow 2's input role. Users that want to find workflows that take in 3D structures will judge workflows that produce them as final output irrelevant.

Choosing a model of similarity. Which similarity approach and metrics are most suited for a given workflow discovery context is not obvious. When is it enough to calculate the number of operations shared and not shared between workflows versus when does one need to carefully distinguish between the role of inputs and outputs? We do not provide an answer to these questions. Rather, we develop *conditions to specify identity between different workflow features* to help answer them. The conditions relate to the two discussed cognitive models of similarity as follows:

- In terms of the featural approach to similarity, the conditions specify when particular features and groups of features are common between workflows. Our features correspond to workflow elements and combinations thereof, as introduced earlier in section 2.1.3. The more elements are identical between workflows, the more similar they are. When the conditions are not satisfied, they are informative of the distinctive features between workflows, as needed to calculate the Tversky measure. The conditions can serve as a basis for exploring a featural approach.

- The conditions capture the structure contained within workflows. We specify basic conditions to describe when and how workflows share elements at different levels. They provide the building blocks for an alignment-based approach.

In summary, we claim that the identity conditions developed in the following subsections will help determine the suitable models of similarity for different workflow discovery tasks. We investigate identity-based matching at the parameter, operation and overall workflow level.

Identity matching at the parameter level

The parameter is at the lowest level in the workflow definition of section 2.1.3. We define the polymorphic function $SamePar$ to pin down what is meant by identity matching at the parameter level. It says that two parameters of operations are identical if they have the same name, syntactic and semantic type. $SamePar$ is defined as:

1) The case for identity between input parameters.

$SamePar : IN \times IN \longrightarrow Boolean$

where

$$SamePar(p_1, p_2) = \begin{cases} true & \text{if } p_1.name = p_2.name \wedge \\ & p_1.str\_type = p_2.str\_type \wedge \\ & p_1.sem\_type = p_2.sem\_type \\ false & \text{otherwise.} \end{cases}$$

2) The case for identity between output parameters.

$SamePar$: $OUT \times OUT \longrightarrow Boolean$

where

$$SamePar(p_1, p_2) = \begin{cases} true & \text{if } p_1.name = p_2.name \wedge \\ & p_1.str\_type = p_2.str\_type \wedge \\ & p_1.sem\_type = p_2.sem\_type \\ false & \text{otherwise.} \end{cases}$$

3) The case for identity between parameters linked to operations.

$SamePar$: $(OP \times (OUT \cup IN)) \times (\text{OP} \times (\text{OUT} \cup \text{IN})) \longrightarrow \text{Boolean}$

$$SamePar((op_1, p_1), (op_2, p_2)) = \begin{cases} true & \text{if } SamePar(p_1, p_2) = true \\ false & \text{otherwise.} \end{cases}$$

We acknowledge that this is an oversimplification of what constitutes identical parameters:

- Parameter names may use different terminology and structural types to indicate the same concept. Reliance on the compatibility between the semantic types of parameters could be enough to establish identity, in case the semantic descriptions are precise enough. In [WGG+04], the example is given of a BLAST service which is implemented differently by two service providers. The functionality offered by both providers is identical, however.
- A subsumption relationship may exist between types that entails parameter compatibility.

The above conditions can be refined to incorporate these aspects. The literature on service discovery has captured the various matching issues extensively, see *e.g.* [BEP06], [HZB+06] or [SPAS03]. Our focus rather is on characterising matches at the operation and workflow level. The matching types describe at those levels will re-use the

conditions defined at the parameter level.

Identity matching at the operation level

Based on the $SamePar$ definition for parameters, we define identity between operations through the $SameOp$ function for operations.

The function requires that the identifier and location of two operations match. It also requires that there is a bijection between the inputs of the first operation and the inputs of the second operation. Similarly, it requires that a bijection exists between the outputs of the first operation and the outputs of the second operation.

The function $SameOp$ is defined as:

$SameOp$: *(OP* $\times$ *OP)*$\longrightarrow$ *Boolean*

where

$$SameOp(OP_1, OP_2) = \begin{cases} true & \text{if } OP_1.name = OP_2.name \wedge \\ & OP_1.loc = OP_2.loc \wedge \\ & \forall p \in (OP_1.in \cup OP_1.out), \\ & \exists p\prime \in (OP_2.in \cup OP_2.out) \mid \\ & \quad SamePar(p, p\prime) = true \wedge \\ & \forall q \in (OP_2.in \cup OP_2.out), \\ & \exists q\prime \in (OP_1.in \cup OP_1.out) \mid \\ & \quad SamePar(q, q\prime) = true \\ false & \text{otherwise.} \end{cases}$$

Notice how this function relies on the $SamePar$ function and its type restrictions. $SamePar$ accepts pairs of inputs or pairs of outputs, not a mix of inputs and outputs.[8]

One could imagine *less stringent conditions* with respect to operation identity such as removing the need for a shared location, name, or inputs and outputs.

More stringent conditions could be imposed by demanding that the functional relationship between the inputs and outputs is the same for two operations. We do not assume such knowledge is available (nor is there room for it in our adopted definition of a workflow). See the work of [HZB+06] on task descriptions for stateless semantic services that can be reasoned over based on the Web Ontology Language OWL

[8] In the (unlikely) case that duplicate operations exist in a service, the above condition would treat these as separate operations.

[HS03].

Identity matching at the workflow level

Identity matching at the parameter and operation level enables us to identify different types of matches at the workflow level. They depend on the level of detail one takes into account. We look at identity of sets of parameters, identity of sets of operations and identity of sets of operations linked through data links.

1) Identity of parameter sets. Recall that *W.IN* and *W.OUT* represents the sets of inputs and outputs of all operations that are part of a workflow, whereas *W.in* and *W.out* denote the overall input and output parameters of the whole workflow *W*.

$SamePars_{OVERALL}$ is used to define when workflows have the same overall input and output parameters.

$SamePars_{OVERALL}$: *(W × W)⟶ Boolean* with

$$SamePars_{OVERALL}(W_1, W_2) = \begin{cases} true & \text{if } \forall p \in W_1.in \cup W_1.out, \\ & \exists p\prime \in W_2.in \cup W_2.out \mid \\ & \quad SamePar(p, p\prime) = true \wedge \\ & \forall q \in (W_2.in \cup W_2.out), \\ & \exists q\prime \in (W_1.in \cup W_1.out) \mid \\ & \quad SamePar(q, q\prime) = true \\ false & \text{otherwise.} \end{cases}$$

$SamePars_{INTERNAL}$ is used to define when all parameters between workflows are shared.

$SamePars_{INTERNAL}$: *(W × W)⟶ Boolean* with

$$SamePars_{INTERNAL}(W_1, W_2) = \begin{cases} true & \text{if } \forall p \in W_1.IN \cup W_1.OUT, \\ & \exists p\prime \in W_2.IN \cup W_2.OUT \mid \\ & SamePar(p, p\prime) = true \wedge \\ & \forall q \in (W_2.IN \cup W_2.OUT), \\ & \exists q\prime \in (W_1.IN \cup W_1.OUT) \mid \\ & SamePar(q, q\prime) = true \\ false & \text{otherwise.} \end{cases}$$

2) Identity of operation sets.

To define when two workflows have the same operations, we introduce $SameOps$.

$SameOps$: *(W × W)⟶ Boolean* with

$$SameOps(W_1, W_2) = \begin{cases} true & \text{if } \forall op \in W_1.OP, \\ & \exists op\prime \in W_2.OP \mid \\ & SameOp(op, op\prime) = true \wedge \\ & \forall oq \in W_2.OP, \\ & \exists oq\prime \in W_1.OP \mid \\ & SameOp(oq, oq\prime) = true \\ false & \text{otherwise.} \end{cases}$$

3) Identity of operation sets linked through data links. Workflows also define links between the operations. To include these in the definition of identity, we rely on $SameW$. Data link correspondence is imposed by requiring the respective data link sets of the workflows are the same.

$SameW$: *(W × W)⟶ Boolean* with

$$SameW(W_1, W_2) = \begin{cases} true & \text{if } W_1\text{.name} = W_2\text{.name} \\ & SameOps(W_1, W_2) = true \wedge \\ & W_1.DL = W_2.DL \\ false & \text{otherwise.} \end{cases}$$

The inclusion of data links and name equality brings us to a *definition of workflow*

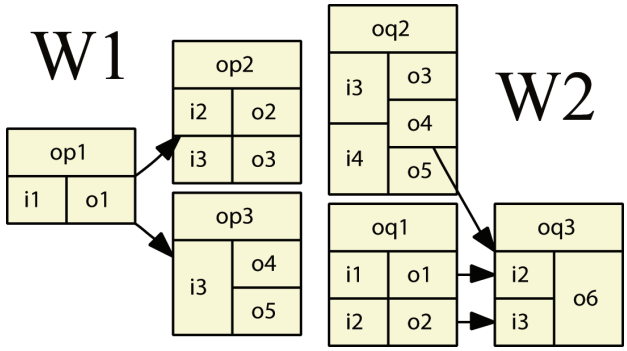


Figure 3.8: Data flow workflow examples repeated.

identity for data flow workflows. We define two workflows to be the same if they have the same operations and the same data links combining those operations. Note that this can be made weaker by not requiring their names are equal.

This concludes our discussion of identity-based matching types for workflows. As we will show in section 3.5, the defined set of functions allows to characterise discovery tasks that support making replacements, insertions and extensions that involve *overlapping* workflow fragments.

3.4.4 Complement-based matching

For discovery tasks where complementarities between workflows are sought (for example, to find extensions *without overlaps* between the workflows), a class of matching types is needed that does not rely on finding commonalities. Although, at the lowest level, finding complementarities also involves establishing identity, we keep similarity-based and complement-based matching clearly separated.[9]

We follow the organisation of the previous section by discussing matching on the parameter level, the operation level and finally the workflow level. We use the workflows in Figure 3.8 (repeated from page 34) as an example.

Complement matching at the parameter level

We start with the notion of a complement match at the level of a parameter. It specifies that an output parameter complements an input parameter if they correspond on the elements of the definition of a parameter. The function $ComplPar$ is defined as:

[9]Note that *complement* here is being used in a different way to the way it is used in the OWL community, where it is taken to refer to negation.

$ComplPar$: *(OUT* $\times$ *IN)* $\longrightarrow$ *Boolean*
where

$$ComplPar(p_1, p_2) = \begin{cases} true & \text{if } p_1.name = p_2.name \wedge \\ & p_1.str\_type = p_2.str\_type \wedge \\ & p_1.sem\_type = p_2.sem\_type \\ false & \text{otherwise.} \end{cases}$$

Taking the example of Fig. 3.8, which parameters are to be compared depends on the use case. If for example a user would like to combine service operation $op2$ with service operation $oq1$, then pairwise comparisons of output and input parameters would be performed.

Akin to the case of parameter identity considered in section 3.4.3, this is a serious simplification. We refer back to that section for possible extensions. Our focus is again on defining matching types higher-up in the workflow element stack.

Complement matching at the operation level

$ComplPar$ helps to define a series of match types relating to the operation level of a workflow description. We distinguish between exact match, super match, sub match and overlap match.

Exact match The $ComplOp_{EXACT}$ function specifies conditions to have an exact match based on two complementary operations. This definition of match type is very strict. For two operations to be compatible, the function requires that all outputs of one operation are compatible with at least one input of a second operation, and that all inputs of that second operation are compatible with at least one output of the first operation. The function $ComplOp_{EXACT}$ is defined as:

$ComplOp_{EXACT}$: *(OP* $\times$ *OP)*$\longrightarrow$ *Boolean*
where

$$ComplOp_{EXACT}(OP_1, OP_2) = \begin{cases} true & \text{if } \forall p \in (OP_1.out \cup OP_2.in), \\ & \exists p\prime \in (OP_1.out \cup OP_2.in) \mid \\ & (p \in OP_1.out \wedge ComplPar(p, p\prime)) \vee \\ & (p \in OP_2.in \wedge ComplPar(p\prime, p)) \\ false & \text{otherwise.} \end{cases}$$

Note that we rely on the typing restriction for inputs in the above $ComplPar$ function to help express the desired semantics.

Going back to the example of Fig. 3.8, to combine service operation $op2$ and service operation $oq1$, pairwise comparisons of all output parameters of $op2$ and input parameters of $oq1$ would need to be performed. Given that parameter names are not equal in any of the cases, there is no exact match.

The Graves disease workflow developed by Peter Li [LHJ+04] for example contains mostly exact matches between operations. The reason for these perfect matches is that its constituent services were written to match together. Most bioinformatics workflows are assembled from distributed autonomous sources, however. It is rare that all output parameters of an operation need to be linked to a follow-up operation for their combination to make sense. Similarly, not all input parameters of an operation necessarily need to receive input for the operation to work. See for example the workflows developed in [STW+04] and [FHW+07]. Alternative, more flexible, definitions of what consists complementary operations are therefore needed. We define these alternatives by introducing relaxations into the definition for an exact match. We re-use the $ComplPar$ function to define what compatibility at the parameter level means.

Bar the empty match, the previous example based on Fig. 3.8 satisfies none of these more relaxed match conditions, because $ComplPar$ stipulates that complementary parameters need to have equal names.

Super match Not all inputs of the second operation need to receive input from the first operation;

$$ComplOp_{SUPER}(OP_1, OP_2) = \begin{cases} true & \text{if } \forall p \in OP_1.out, \\ & \exists p\prime \in OP_2.in \mid \\ & ComplPar(p, p\prime) = true \\ false & \text{otherwise.} \end{cases}$$

Sub match Not all outputs of the first operation need to match an input of the second operation;

$$ComplOp_{SUB}(OP_1, OP_2) = \begin{cases} true & \text{if } \forall p \in OP_2.in, \\ & \exists p\prime \in OP_1.out \mid \\ & ComplPar(p\prime, p) = true \\ false & \text{otherwise.} \end{cases}$$

Overlap match Not all inputs of the second operation need to receive input from the first operation and not all outputs of the first operation need to match an input in the second operation;

$$ComplOp_{OVERLAP}(OP_1, OP_2) = \begin{cases} true & \text{if } \exists (p, p\prime) \in (OP_1.out \times OP_2.in) \mid \\ & ComplPar(p, p\prime) = true \\ false & \text{otherwise.} \end{cases}$$

Empty match Between the operations no inputs and outputs match;

$$ComplOp_{EMPTY}(OP_1, OP_2) = \begin{cases} true & \text{if } \nexists (p, p\prime) \in (OP_1.out \times OP_2.in) \mid \\ & ComplPar(p, p\prime) = true \\ false & \text{otherwise.} \end{cases}$$

Two remarks should be made at this point:

- We mentioned earlier that the $ComplPar$ function of complementarity between parameters covers just one case. In the interest of brevity, we do not elaborate on the interplay of the other cases with matching at the operation level.
- We pointed out that different workflows satisfy different operation compatibility conditions. Fulfilling $ComplOp_{EXACT}$ is the right condition to impose to identify compatibility in one case, but it is too strict when considering other cases. Conversely, more loose conditions such as $ComplOp_{OVERLAP}$ risk being too imprecise. Conditions that incorporate knowledge of *cardinality constraints* of operation inputs would reduce this uncertainty and better predict when two operations combinations are compatible [BEP06]. However, since workflows operations that are published as part of a Web service in the WSDL language

typically do not indicate which cardinality constraints apply to their inputs, additional annotation would need to be provided to use cardinality as a basis for comparison.

Complement matching at the workflow level

The definitions of matching at a parameter and operation level allow to define what a match entails at a workflow level. Again, multiple choices are available: exact match, super match, sub match, overlap match and empty match.

It is important to note that only a subset of operations in OP is relevant to the matching context at hand, *i.e.* only the relevant "fringe" of the workflows should be considered. User selection or automated workflow analysis should determine the two sets of operations relevant to the matching problem at hand.

For example, say a user wants to couple workflows 1 and 2 in Figure 3.8. Let us assume the user is only interested in operations op_2 and op_3 from W_1 and oq_1 and oq_2 from W_2. We define $W_x.OP_{FR}$ as the *set of operations indicating fringe* x for a workflow matching problem. In the particular example above, the two relevant sets of operations are:

$W_1.OP_{FR}$ = $\{op_2, op_3\}$ and
$W_2.OP_{FR}$ = $\{oq_1, oq_2\}$.

Different degrees of match are possible between $W_1.OP_{FR}$ and $W_2.OP_{FR}$.

Exact match The function $ComplW_{EXACT}(W_1, W_2)$ is defined as:

$$ComplW_{EXACT}(W_1, W_2) = \begin{cases} true & \text{if } \forall op \in (W_1.OP_{FR} \cup W_2.OP_{FR}), \\ & \exists op\prime \in (W_1.OP_{FR} \cup W_2.OP_{FR}) \mid \\ & (op \in W_1.OP_{FR} \wedge ComplOp_{EXACT}(op, op\prime)) \vee \\ & (op \in W_2.OP_{FR} \wedge ComplOp_{EXACT}(op\prime, op)) \\ false & \text{otherwise.} \end{cases}$$

The definition is very strict. It requires that all operations in the fringe of the first workflow match with operations in the fringe of the second workflow. When they match, they should do so on all parameters, as defined by the $ComplOp_{EXACT}$ function.

Note that the above example easily fails to satisfy the condition, since $ComplOp_{EXACT}$

relies on $ComplPar$ at the parameter level. $ComplPar$ requires that names of parameters are equal, which is never the case in the example.

Similar to the previous section, different uses of the qualifiers lead to more loose definitions of what consists a match between complementary operations. For now, we continue to rely on the semantics of the $ComplOp_{EXACT}$ function.

Super match Not all operations in the fringe of the second workflow need to be compatible with an operation in the fringe of the first workflow;

$$ComplW_{SUPER}(W_1, W_2) = \begin{cases} true & \text{if } \forall op \in W_1.OP_{FR} \\ & \exists op\prime \in W_2.OP_{FR} \mid \\ & ComplOp_{EXACT}(op, op\prime) = true \\ false & \text{otherwise.} \end{cases}$$

The above example fails since at least one operation needs to match between the two fringes.

Sub match Not all operations in the fringe of the first workflow need to match an operation in the fringe of the second workflow;

$$ComplW_{SUB}(W_1, W_2) = \begin{cases} true & \text{if } \exists (op, op\prime) \in (W_1.OP_{FR} \times W_2.OP_{FR}) \mid \\ & ComplOp_{EXACT}(op, op\prime) = true \\ false & \text{otherwise.} \end{cases}$$

In this case, again the example fails since no operations match between the fringes.

Overlap match Not all operations in the fringe of the second workflow need to match an operation in the fringe of the first workflow and not all operations in the fringe of the first workflow need to match an operation in the fringe of the second workflow;

$$ComplW_{OVERLAP}(W_1, W_2) = \begin{cases} true & \text{if } \exists (op, op\prime) \in (W_1.OP_{FR} \times W_2.OP_{FR}) \mid \\ & ComplOp_{EXACT}(op, op\prime) = true \\ false & \text{otherwise.} \end{cases}$$

Similar to the above cases, at least one operation needs to match hence there is no overlap match between the workflows in the example.

Empty match Between the workflows no operations from the respective fringes match;

$$ComplW_{EMPTY}(W_1, W_2) = \begin{cases} true & \text{if } \nexists (op, op\prime) \in (W_1.OP_{FR} \times W_2.OP_{FR}) \mid \\ & ComplOp_{EXACT}(op, op\prime) = true \\ false & \text{otherwise.} \end{cases}$$

Since no operations match, there is an empty match between the workflow fringes in the example.

Each of the match definitions can be made even more loose by substituting the chosen $ComplOp_{EXACT}$ operation match function. $ComplOp_{SUPER}, ComplOp_{SUB}$, $ComplOp_{OVERLAP}$ and $ComplW_{EMPTY}$ all provide additional matching criteria. In total, there are five workflow level match definitions with each having five possibilities at the operation matching level. For readability purposes, we do not include all definitions. If multiple parameter compatibility criteria were adopted, the number would explode further.

Practically speaking, the use of $ComplOp_{EMPTY}$ is only meaningful as a vehicle for rewriting $ComplW_{EMPTY}$. It is not sensible to speak of a match at the workflow level when no parameters match at the operation level.

We conclude our analysis of workflow matching types by considering the question of *how to choose between the exact, super, sub, overlap and empty variants of the matching types.* We gave examples on page 71 of how different styles of building workflows lead to different variants being relevant. Which variant is the most appropriate for a given workflow discovery task therefore depends on the application domain. Analysis of practical re-use cases should yield heuristics to guide the choice. Chapter 4 collects a benchmark re-use cases solved by bioinformaticians. It could serve as the

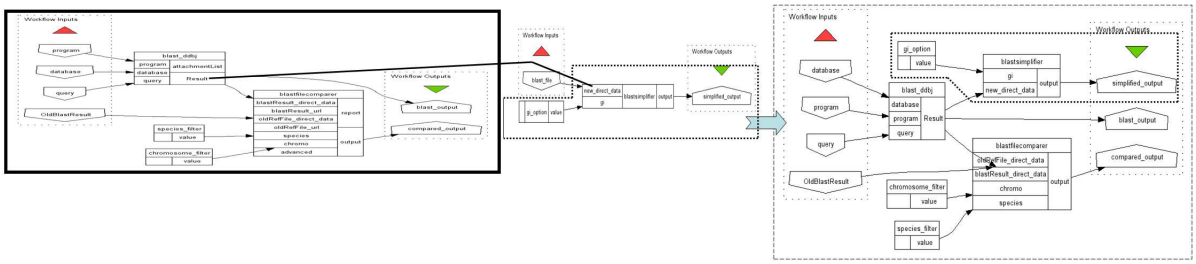


Figure 3.9: Appending data flows.

basis for such an analysis. This in turn could guide the design of discovery tools.

3.5 Workflow discovery tasks formally

In the previous chapter we gave a formal definition of a data flow workflow. Here, we added (i) a characterisation of the role of workflow discovery for workflow re-use and repurposing and (ii) an analysis of the different matching types that apply to workflow discovery.

Together, these allow for a formal description of the discovery tasks. They are to support finding adequate extensions, insertions and replacements for a given workflow. We provide examples of each in Fig. 3.9, 3.10, 3.11 and 3.12. They show two extensions (one by prepending a workflow, the other by appending), one insertion and one replacement. The dotted box indicates the workflow fragment that is to be combined with a given, original workflow (indicated with a black box). The result of their combination is shown, too.

In what follows we provide definitions for workflow discovery tasks that:

- support re-use through calculating workflow similarity and
- support repurposing by defining when one workflow extends another, when one can be inserted in the other and when one can replace the other.[10] Discovery techniques can exploit these conditions to assess whether a particular relationship holds between a given workflow and workflows available from a pool. For

[10]Obviously there is no discovery task that supports making a deletion to a workflow.

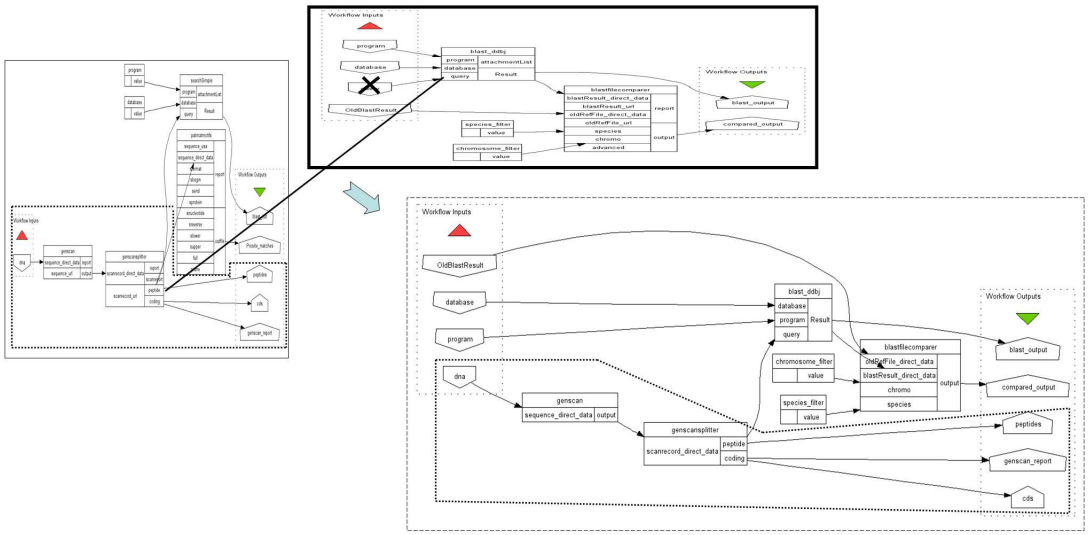


Figure 3.10: Prepending data flows.

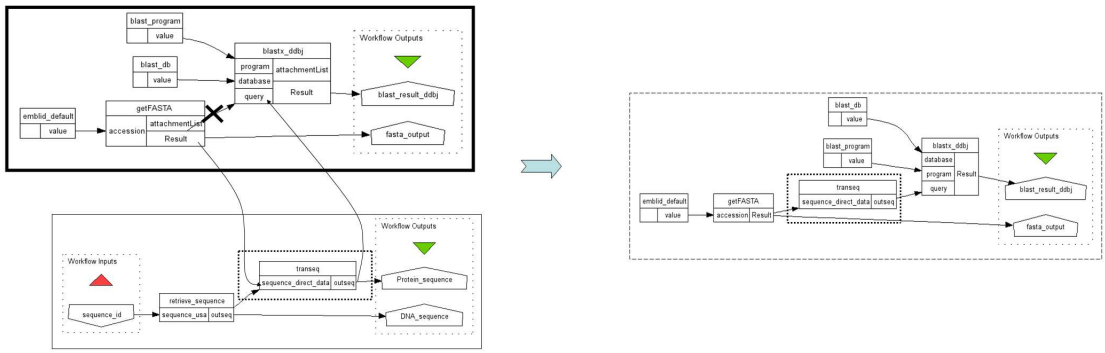


Figure 3.11: Insertion in data flows.

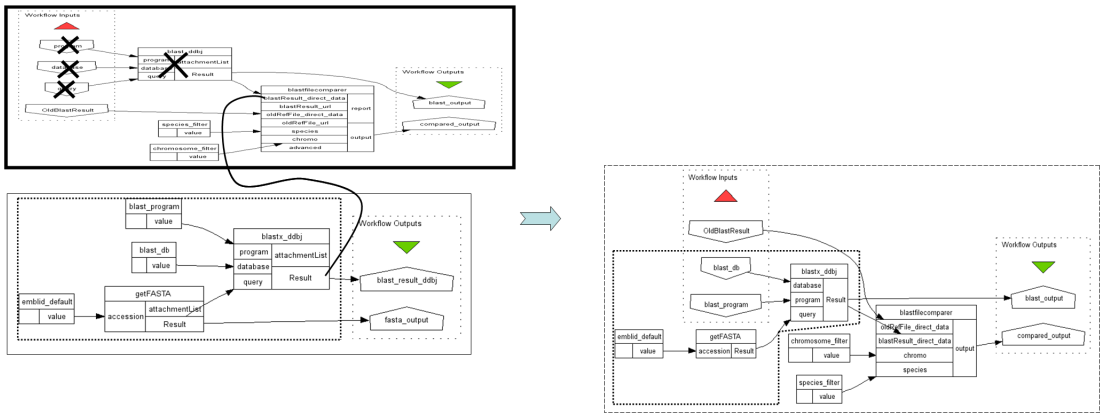


Figure 3.12: Replacement in data flows.

each repurposing scenario, we define exact and overlapping matches. The notion of overlap will require the combination of measures of complementarity and similarity.

3.5.1 Calculating workflow similarity

On page 63 we summarised the major similarity paradigms in use in the cognitive sciences. Having defined data flow workflow matching types in the previous section, one can specify a suite of similarity metrics based on these paradigms. We provide but one example here; metrics can be defined for each of the paradigms. Their definition (and evaluation) is work for the future.

Recall the example of a featural metric, the Tversky measure:

$$S(A, B) = \theta f(A \cap B) - \alpha f(A - B) - \beta f(B - A)$$

Say A equals $W_1.OP$, the set of operations in workflow W_1 and B equals $W_2.OP$, the set of operations in workflow W_2. We need to determine which operations are shared and not shared between W_1 and W_2. Testing for the condition $SameOp$ on the cross product of the two sets helps to obtain the required data.

3.5.2 Finding workflow extensions

A workflow extends another one either by appending it or by prepending it.[11] We distinguish between extensions that are based on matching the outer ends of the workflows involved and those that consider overlaps between workflows, *i.e.* when two workflows share a workflow fragment.

We rely on the earlier specified match conditions to define when workflows extend each other. This approach casts the notion of an extension in terms of whether or not particular match conditions are met. It provides an unambiguous way to document what is meant by an extension.

Exact match Workflow W_2 *exact-appends* workflow $W_1 \Longleftrightarrow$

$$AppendsW_{EXACT}(W_2, W_1) = true$$

where $AppendsW_{EXACT}$: $(W \times W) \longrightarrow$ *Boolean* with

$$AppendsW_{EXACT}(W_2, W_1) = \begin{cases} true & \text{if } ComplW_{EXACT}(W_1, W_2) = true \\ false & \text{otherwise.} \end{cases}$$

Workflow W_2 *exact-prepends* workflow $W_1 \Longleftrightarrow$

$$PrependsW_{EXACT}(W_2, W_1) = true$$

where $PrependsW_{EXACT}$: $(W \times W) \longrightarrow$ *Boolean* with

$$PrependsW_{EXACT}(W_2, W_1) = \begin{cases} true & \text{if } ComplW_{EXACT}(W_2, W_1) = true \\ false & \text{otherwise.} \end{cases}$$

Both conditions are strict and assume the workflows fit together perfectly. Changing the $ComplW$ workflow matching function with any of the alternative functions introduced in the previous section would make them more relaxed.

[11] One additional, interesting, case of an extension (but not treated here) takes into account the introduction of additional data links.

Overlap match In Figure 3.8 (on page 69), only operations $op1, op3, oq1$ and $oq2$ are of relevance for establishing whether the two workflows *exact-extend* each other. In an alternative scenario, one workflow may extend another through an "overlap." We define the auxiliary function $OverlapsW$, which relies on the $ContainsW$ function (defined earlier on page 35).

$OverlapsW$: *(W* $\times$ *W* $\times$ *W)*$\longrightarrow$ *Boolean*

with

$$OverlapsW(FW_1, W_1, W_2) = \begin{cases} true & \text{if } ContainsW(W_1, FW_1) = true \wedge \\ & ContainsW(W_2, FW_1) = true \\ false & \text{otherwise.} \end{cases}$$

Based on $OverlapsW$, we define when a workflow appends another in case the two overlap.

Workflow W_2 *overlap-appends* workflow $W_1 \Longleftrightarrow$

$$AppendsW_{OVERLAP}(W_2, W_1) = true$$

where $AppendsW_{OVERLAP}$: *(W* $\times$ *W)*$\longrightarrow$ *Boolean* with

$$AppendsW_{OVERLAP}(W_2, W_1) = \begin{cases} true & \text{if } \exists\ (FW_1, FW_2) \mid \\ & OverlapsW(FW_1, W_1, W_2) = true \wedge \\ & ContainsW(W_2, FW_2) = true \wedge \\ & CompW_{EXACT}(FW_1, FW_2) = true \\ false & \text{otherwise.} \end{cases}$$

Observe how the notion of overlap means we need to combine measures of complementarity and similarity. In a similar vein, we define when a workflow prepends another one in case they overlap.

Workflow W_2 *overlap-prepends* workflow $W_1 \Longleftrightarrow$

$$PrependsW_{OVERLAP}(W_2, W_1) = true$$

where $PrependsW_{OVERLAP}$: *(W* $\times$ *W)*$\longrightarrow$ *Boolean* with

$$PrependsW_{OVERLAP}(W_2, W_1) = \begin{cases} true & \exists\ (FW_1,\ FW_2)\ | \\ & OverlapsW(FW_1,\ W_1,\ W_2) = true \wedge \\ & ContainsW(W_2,\ FW_2) = true \wedge \\ & CompW_{EXACT}(FW_2,\ FW_1) = true \\ false & \text{otherwise.} \end{cases}$$

3.5.3 Finding workflow insertions

In a similar fashion, we define workflow insertion for the exact case and for the overlap case.

Exact match Workflow W_3 *exact-inserts* workflows W_1 and $W_2 \iff$

$$InsertsW_{EXACT}(W_3, W_1, W_2) = true$$

where $InsertsW_{EXACT}$: $(W \times W \times W) \longrightarrow$ *Boolean* with

$$InsertsW_{EXACT}(W_3, W_1, W_2) = \begin{cases} true & \text{if } AppendsW_{EXACT}(W_3,\ W_1) = true \wedge \\ & PrependsW_{EXACT}(W_3,\ W_2) = true \\ false & \text{otherwise.} \end{cases}$$

Overlap match Workflow W_3 *overlap-inserts* workflows W_1 and $W_2 \iff$

$$InsertsW_{OVERLAP}(W_3, W_1, W_2) = true$$

where $InsertsW_{OVERLAP}$: $(W \times W \times W) \longrightarrow$ *Boolean* with

$$InsertsW_{OVERLAP}(W_3, W_1, W_2) = \begin{cases} true & \text{if } AppendsW_{OVERLAP}(W_3,\ W_1) = true \wedge \\ & PrependsW_{OVERLAP}(W_3,\ W_2) = true \\ false & \text{otherwise.} \end{cases}$$

3.5.4 Finding workflow replacements

Finally, we look at workflow replacements, again for the exact case and for the overlap case.

Exact match Workflow W_2 *exact-replaces* workflow $W_1 \iff$

$$ReplacesW_{EXACT}(W_2, W_1) = true$$

where $ReplacesW_{EXACT}$: $(W \times W \times W) \longrightarrow$ *Boolean* with

$$ReplacesW_{EXACT}(W_2, W_1) = \begin{cases} true & \text{if } SameW(W_1, W_2) = true \\ false & \text{otherwise.} \end{cases}$$

Overlap match Workflow W_2 *overlap-replaces* workflow $W_1 \iff$

$$ReplacesW_{OVERLAP}(W_2, W_1) = true$$

where $ReplacesW_{OVERLAP}$: $(W \times W \times W) \longrightarrow$ *Boolean* with

$$ReplacesW_{OVERLAP}(W_2, W_1) = \begin{cases} true & \text{if } \exists\, FW_2 \mid \\ & SameW_{EXACT}(FW_2, W_1) = true \wedge \\ & ContainsW(W_2, FW_2) \\ false & \text{otherwise.} \end{cases}$$

3.6 Related work

We investigate both workflow discovery solutions deployed in workflow systems and solutions proposed in the research literature and classify them against the matching types developed in this chapter.

3.6.1 Scope

Workflow discovery potentially encompasses a big area of research. We scope the related work section by clarifying which aspect of the workflow lifecycle and particular

tasks we are looking to support. We also highlight the communities that have produced particularly relevant work to date.

Workflow lifecycle context

We highlighted several possible user contexts to base retrieval of workflows on such as wet lab protocols, publications or the social network of a scientist. We focus here solely on the context offered by the workflows themselves.

The reader will remember that workflows have different phases in their lifecycle (repeated below). In each phase of their lifecycle, workflows are associated with different types of information, all of which yield distinct contexts for the retrieval of *concrete* workflows (i.e. the end product of phase 2). For each of the phases we give a simple illustration:

1. During design, while the workflow is still being designed. The *preliminary design* of a workflow can guide a workflow developer to earlier concrete workflows modelled on a similar design.
2. Post design, pre-enactment, as either a finished, concrete workflow where the required resources are known, or as a finished yet abstract workflow (also known as a template) whose resources still will need to be decided dynamically during enactment. An abstract workflow can serve as input to find a cluster of related concrete workflows. *Elements of a concrete workflow* can serve to find related concrete workflows. A single service can act as a basis to retrieve relevant workflows. Likewise, a selection of a subset of services, void of any control flow, can suffice. Sometimes a workflow fragment or the complete concrete workflow will be relevant, including its control flow.
3. During enactment, when intermediary results come about. *Partial results of a long-running workflow* can direct a workflow designer to find concrete workflows that can work off these results.
4. Post enactment, when all results are available. *Quality of service data* collected from enacted workflows can be used for discovery of concrete workflows.

We are purely interested in discovering concrete workflows based on (elements of) concrete workflows and center the related work section around this.

Workflow similarity and complementarity

Communities that adopt workflow re-use are soon faced with the discovery aspect. This chapter argued that, to support *workflow re-use and repurposing* through discovery, there is a need for techniques that assess how similar and complementary workflows are. Many techniques have been developed for component discovery and composition which are potentially relevant.

For *workflow re-use*, we specified formal conditions for similarity of parameters, operations and workflows in the case of data flows. The survey will compare discovery techniques in terms of these conditions.

For *workflow repurposing*, we need to assess how complementary workflows are. We specified formal conditions for complementarity of parameters, operations and workflows in the case of data flows. The survey will compare composition techniques in terms of these conditions. We stated earlier that we only wish to support repurposing as semi-automated composition, given the hands-on approach of most scientists towards building workflows. This should not exclude automated composition techniques from our survey. They are potentially useful (i) as a source of inspiration for semi-automated techniques to establish complementarity between workflow elements and (ii) as a generator of candidate workflow compositions which scientists can select from.

Relevant communities

In general, software components can be *described* by means of their input and output, and/or based on their behaviour, *e.g.* via pre- and postconditions or Finite State Automata. Based on such descriptions, components can be *discovered, composed, configured, verified, simulated, invoked* and *monitored*. Of these, discovery and composition are the most relevant for our purposes. Depending on the formalism and the level of detail used to describe workflows, different techniques apply. A full survey of what has been done historically is beyond our scope. Several communities have produced techniques that support the retrieval of software components, especially *the software engineering, knowledge engineering, formal methods and information retrieval communities*.

Workflows orchestrate both software components that live on the Web as Web services as well as other types of components. At an abstract level, workflows can be regarded as composite components, which means some of the machinery developed

for software component discovery and composition is applicable. The techniques considered in this chapter are selected based on their particular relevance to workflow discovery and workflow composition, as opposed to discovery and composition of *atomic* components. In other words, we exclude work that only considers atomic components as the basis for either a discovery or composition process. We point the interested reader to a survey of discovery techniques for atomic components, published as Web services, by [TIR+07]. Surveys that consider composition of two atomic components, published as Web services, include [DS05] and [RS04]. These surveys do not consider the situation where composite components or *groups* of atomic components are to be matched with other composite components. Our contribution is to present and compare techniques addressing the latter situation. They are from the scientific workflow community, the Web services community, the multi-agent systems community and the workflow community.

3.6.2 Discovery support in scientific workflow systems

We start the survey by determining the level and type of workflow discovery support available in both scientific workflow systems and (covered much more briefly) in business workflow systems. The survey is to reveal a serious lack of support, which will lead us to present techniques developed in other communities.

How does the scientific workflow systems community approach the workflow discovery problem? For commercial systems, information on workflow discovery approaches is sparse. Little information is available from commercial vendors (e.g. the above mentioned Inforsense and SciTegic). Similarly, for business-oriented workflow systems, the above-mentioned survey of [MHH] does not reveal any details on this point. Our impression is that most vendors treat sub-workflows as just another atomic component in the workflow. For this reason we speculate that discovery is mostly performed based on information captured in a sub-workflow's signature and is not based on its structure.

Given the lack of available information, henceforth only work from the academic community will receive attention.

Discovery support in terms of workflow lifecycle phase

Based on the various workflow phases introduced earlier, we provide a high level overview of the discovery support offered by a large section of systems. Both support in scientific workflow systems and standalone provenance management systems is assessed. The presented sample is drawn from the surveys of [YB05], [GSLG05] and [Zha07]. An discussion of the systems presented in Table 3.2 is available from the Provenance Challenge site.[12]

Tables 3.1 and 3.2 reveal that no one system offers support for discovery for all phases. It also shows that the majority of systems offer support for retrieval of at least one type of workflows but sometimes there is none. Depending on whether the onus is on supporting design, enactment (including scheduling of resources) or results management, different functionality has been made available. A detailed discussion of the support available in each category is outside our scope. We will zoom in on the specific functionality systems offer in terms of phase 2.

Discovery support for concrete workflows

Taking the subset of systems that provide some level of support for discovery of concrete workflows, we observe from Table 3.3 that most systems offer no facility to search workflows based on their structure. Workflows are commonly treated as just another service type which can be retrieved by its signature description. Exceptions are the VisTrails system [SVK+07] and Chimera [ZWF06]. We discuss both systems in more detail in the next section.

3.6.3 Techniques in support of concrete workflow discovery

Besides the techniques available in operational e-science systems, a range of relevant research prototypes have been developed in other communities. We distinguish between techniques that focus specifically on workflow discovery and work that tackles (semi-)automated workflow composition. The next section will then classify a subset of techniques (those capable of assessing workflow similarity and complementarity) in terms of the workflow matching conditions we defined earlier.

Workflow system	Workflow lifecycle phase 1	Phase 2	Phase 3	Phase 4
VisTrails [SVK+07]	yes	yes	no	yes
Taverna [OGA+05]	no	yes	no	yes
Kepler [LAB+05]	no	yes	no	yes
Chimera [ZWF06]	no	yes	no	yes
JOpera [PA04]	no	yes	no	yes
Triana [MWG04]	no	yes	no	no
Geodise [TCS+04]	no	yes	no	no
Sedna [WEB+07]	no	yes	no	no
WINGS [GRD+07]	no	yes	no	no
Pegasus [DBG+04]	no	no	yes	no
DAGMan [TWML01]	no	no	yes	no
Askalon [WPF05]	no	no	yes	no
Gridbus Broker [SVW06]	no	no	yes	no
GrADS [Gro07b]	no	no	no	no
GridFlow [Gro07c]	no	no	no	no
ICENI [MYA+04]	no	no	no	no
Karajan [Las05]	no	no	no	no
UNICORE [For04]	no	no	no	no

Table 3.1: Support in scientific workflow systems for workflow discovery, based on the phase in the workflow lifecycle.

Standalone provenance systems	Workflow lifecycle phase 1	Phase 2	Phase 3	Phase 4
Pasoa	no	no	no	yes
REDUX	no	no	no	yes
Karma	no	no	no	yes
PASS	no	no	no	yes
CESNet	no	no	no	yes
Zoom	no	no	no	yes
ES3	no	no	no	yes

Table 3.2: Support in scientific provenance systems for workflow discovery, based on the phase in the workflow lifecycle.

Workflow system	Discovery based on signature	Discovery based on structure
VisTrails [SVK+07]	yes	yes
Chimera [ZWF06]	yes	yes
Taverna [OGA+05]	yes	no
Kepler [LAB+05]	yes	no
JOpera [PA04]	yes	no
Triana [MWG04]	yes	no
Geodise [TCS+04]	yes	no
Sedna [WEB+07]	yes	no
WINGS [GRD+07]	yes	no

Table 3.3: Support in scientific workflow systems for workflow discovery in phase 2, based on workflow signature and structure.

Concrete workflow discovery techniques

Table 3.4 provides an overview of approaches in support of workflow discovery. It makes clear that multiple workflow languages are being investigated and that the scope of each approach is limited to only one language. Different data structures are in play to represent workflows, with graphs being a popular option, and different techniques work over these structures. The table also reveals that none of the listed techniques support the discovery of complementary workflows based on "bridging" workflow fringes (cfr. page 69) - such capability is more commonplace in workflow composition techniques (see the next section).

The Chimera system, [ZWF06] and introduced above, translates workflows available as Virtual Data Language specifications into untyped graphs. The system allows retrieval of workflows by example, in that a query graph can be fed into the system in order to retrieve pipelines extending the one represented by the query graph.

The earlier mentioned VisTrails e-science system [SVK+07] enables querying of pipelines of specialised visualization modules from the VTK dataflow-based visualization system. It translates the pipelines into typed graphs and relies on a graph matcher to offer exact (pattern-based) and approximate search. Akin to Chimera, VisTrails relies on a graph matcher for pattern-based search. This permits to retrieve pipelines by example. In addition, it allows for approximate search. This is implemented by first

[12]Web site: `http://twiki.pasoa.ecs.soton.ac.uk/bin/view/Challenge/ParticipatingTeams`

Reference	Workflow language	Data structure	Technique	Similarity	Complementarity
Chimera [ZWF06]	VDL	Untyped graph	Exact graph matching	no	no
VisTrails [SVK+07]	VTK	Typed graph	(In-)exact graph matching	yes	no
[BEKM06]	BPEL	Typed graph	Exact graph matching	no	no
[CGB06]	BPEL	Typed graph	Inexact graph matching	yes	no
[BK02]	MIT Process Handbook (PQL)	Typed graph	PQL	no	no
[KBL+07]	MIT Process Handbook (OWL)	Typed graph (RDF)	iSPARQL	yes	no
myGrid [WSG+03]	SCUFL	Description Logic (OWL) concepts	Description Logic (OWL) classification	no	no
[MW06]	RosettaNet PIP	Annotated FSM	N-gram indexing	yes	no

Table 3.4: Techniques for automated workflow discovery.

calculating how one pipeline can be transformed into another and then recording the relative ease with which this can be done between all possible pipeline pairs.

Beeri and colleagues [BEKM06] developed and implemented a graph-based query language for BPEL (Business Process Execution Language) workflows. Following the same logic of the previous paragraph, the approach could be used for retrieval of workflow extensions. While also relying on the representation of a BPEL workflow as a graph, Corrales et al. [CGB06] (prototype available on-line[13]) instead use error-correcting graph subisomorphism detection. The technique enables them to calculate an edit distance between graphs and hence to define a structural similarity metric for workflows.

Contrasting the work of Bernstein and Klein [BK02] with that of Kiefer et al. [KBL+07] yields a similar "exact versus inexact" querying capability. Bernstein designed a query language (the Process Query Language) to enable exact structural queries over an Entity Relationship (ER) diagram. The ER diagram in question formalises the structure of processes available in the MIT Process Handbook. Kiefer on the other hand also worked over Process Handbook entries, but translated these into a graph specification (in RDF) and relied on text similarity in the contents of graph nodes to retrieve similar graphs.

A Description Logic based approach is explored by Wroe and colleagues [WSG+03]. They abstract Taverna workflows, written in the Scufl language, to be a bag of services and discard all structural relationships between these services. The workflows are represented by concepts in DAML+OIL, a precursor of OWL, the Web Ontology Language [HPSvH03]. DAML+OIL is underpinned by a very expressive Description Logic to allow automated logical reasoning. Workflows are represented by concepts that have part-of relationships with other concepts that in turn describe the constituent services of a workflow. Subsumption reasoning is used to detect whether the services in one bag subsume the set of services in another.

Finally, Mahleko and Wombacher [MW06] work over workflows in the form of RosettaNet Partner Interface Processes. Their approach tackles the hard computational complexity the above approaches are typified by, and proposes the use of Finite State Machines (FSMs) to obtain fast performance during matching. The formalisation of a workflow into an FSM can be done with decreasing accuracy. Accuracy decreases gradually as fewer of all possible paths in the control flow of a workflow are encoded by the FSM. A less precise formalisation in turn enables speedier comparisons. By

[13] Web site: `http://ariadna.unicauca.edu.co/BPELService/`

Reference	Workflow language	Data structure	Technique	Similarity	Complementarity
[KGR06]	OWL	Description Logic (OWL) concepts	Description Logic (OWL) reasoning	no	no
[KWJ+04]	Prolog	Prolog clauses	Inductive logic programming and active learning	yes	no
VisTrails [SVK+07]	VTK	Typed graph	Inexact graph matching	yes	yes
[BGL+04]	BPEL	Mealy FSM	Description Logic (ALC) model checking	no	yes
[MWG04]	OWL	Description Logic (OWL) concepts	Planning	no	no
[LR05]	LCC	Prolog clauses	Incidence calculus	yes	yes
[MBE03]	WSFL and XLANG	CSSL specifications	Composability rules	no	yes

Table 3.5: Techniques for automated workflow composition.

comparing the traces of workflows formalised as FSMs, they are able to detect workflow similarity. Practically, the FSMs are serialized as an N-gram (a sub-sequence of N items from a given sequence) where an increase in N enables a more precise workflow representation. Comparisons are performed through N-gram indexing.

Concrete workflow composition techniques

In addition to techniques for concrete workflow discovery, the literature also contains work to (semi-)automatically construct concrete workflows. Three types of approaches can be found: those that assume the existence of a pre-defined template or framework, those that compose unrelated sets of atomic components without relying on a given structure and those that combine both approaches.

With a template. The WINGS system (Workflow INstance Generation and Selection) is a OWL-based framework for creating workflow instances developed at the University of Southern California [KGR06]. Given a workflow template and input file descriptions in the Web Ontology Language OWL [HPSvH03], the system creates valid instantiations of the workflow template. It has been applied to generate large executable workflows in an earthquake science application.

Another example of the template approach can be found in King et al. [KWJ+04]. Their "robot scientist" auto-generates new sets of actions or experiments (workflows) to undertake based on feedback of running earlier experiments. The robot investigates the function of different genes in baker's yeast. Knowledge on biochemistry, encoded as Prolog clauses, and feedback from watching the yeast grow are combined to generate a set of hypotheses concerning the function of the gene in question. It then generates a new experiment that will eliminate as many of the hypotheses as possible. Conceptually this approach differs from the WINGS approach in that the outcome of running one "workflow" links into the generation and execution of a new one.

Without a template. The VisTrails system [SVK+07], which we introduced as a discovery technique above, also allows the semi-automatic creation of dataflows of visualization modules by analogy. The idea is that the steps that were needed to create one particular pipeline potentially help generate others in the future and thus should be stored as a macro or analogy. The macro is applied whenever other pipelines are found that match its starting context. The relevance of the starting context of a candidate pipeline to a macro is quantified as a weighted average between how compatible its modules are and how similar its module neighbourhood is. Module compatibility is determined by comparing how many ports are shared between modules and neighbourhood similarity by comparing topologies of connecting modules.

Berardi and colleagues [BGL+04] make the (rather strong) assumption that services have their behaviour exposed as Mealy Finite State Machines. Under this assumption, synthesis of different services becomes possible by checking composability of FSMs. Technically this is achieved by reducing the problem to checking satisfiability of formulae in the $\mathcal{ALC}$ Description Logic.

Lambert and Robertson [LR05] make the observation that logic-based approaches will not be able to capture all the pertinent system features in a complex world. Rather, the process of agent composition should take the practical performance of earlier compositions in the real world into account. As a contribution to the multi-agent systems

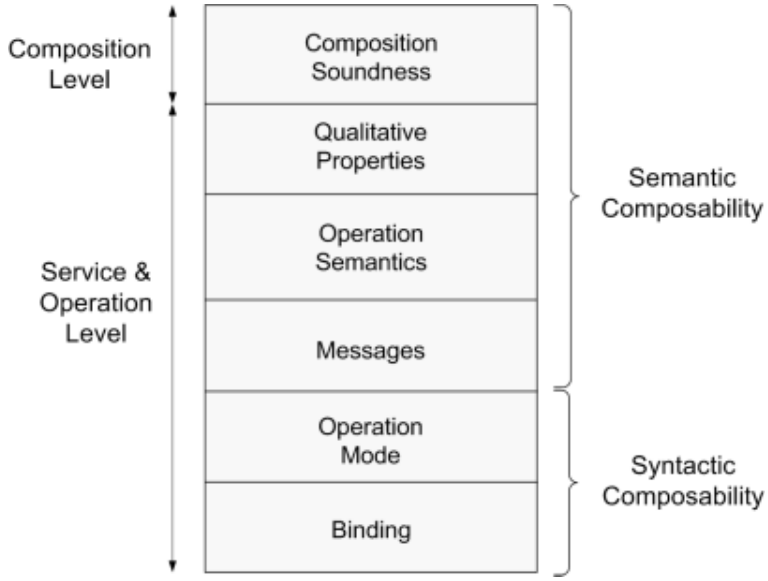


Figure 3.13: Semantic and syntactic service composability.

literature, they develop a statistical approach to agent (*i.e.* service) selection and composition. By relying on the LCC language (Lightweight Coordination Calculus), they gain a protocol for capturing the failure and success of interactions between agents. The protocol's ability to log performance enables them to apply the incidence calculus to gather performance statistics. The thus gathered statistics help inform the agent composition process.

Combining both. Medjahed et al. [MBE03] propose a framework and implement a prototype for the automatic composition of groups of Web services into (WSFL and XLANG) workflows. They rely on a template-based technique and on techniques that work independent of a template. To generate new workflows, the authors propose the use of two sets of *composability rules*. These are to compare the syntactic and semantic properties of Web services (Fig. 3.13):

Syntactic rules They include: (i) mode composability, which compares operation modes (e.g. "one-way," "solicit-response"), and (ii) binding composability, which compares the binding protocols of interacting services [MBE03].

Semantic rules They include: (i) message composability, which compares the number of message parameters, their data types, business roles, and units; (ii) operation semantics composability, which compares the semantics of service operations; (iii) qualitative composability, which compares qualitative properties of Web services; and (iv) composition soundness, which checks whether combining Web

services in a specific way is worthwhile [MBE03].

Composition soundness checks composability at the workflow composition level, unlike the other rules that deal with composability at the Web service and operation levels. Composition soundness is done by checking its compliance against a so-called *composition template*. Composition templates are graphs built using precedence relationships between services, learned by the system or added by experts.

In terms of the matching conditions, the framework of Medjahed is richer than our matching conditions in that it makes more layers of the matching process explicit. It is poorer than ours in that it only considers exact matches and not partial matches (what we described as super matches, overlap matches and empty matches).

3.6.4 Classifying techniques by workflow matching conditions

Not all of the discussed workflow discovery and composition techniques make it possible to assess workflow similarity and complementarity. We classify the ones that do against the workflow matching conditions (see Tables 3.6 and 3.7).

Of those techniques that do similarity assessment, Table 3.6 shows that only VisTrails [SVK+07] allows to meet all conditions. It does not implement the more fine-grained definitions of partial (*i.e.* super and overlap) matches, though. Corrales' approach [CGB06] is able to detect similarities at all workflow levels. For the remaining two techniques, some restrictions are in place. The textual similarity metrics implemented by Kiefer [KBL+07] allow only to rank workflows at a coarse level and do not distinguish between parameters or operations. Partner Interchange Processes, which are the subject of the work of Mahleko [MW06], do not contain inputs or outputs (only activities) and as a result [MW06] provide no facility to make comparisons at the parameter level.

Of the techniques that do complementarity assessment, Table 3.7 shows that the robot scientist work [KWJ+04] does not support the conditions. Similarly, Lambert's statistical technique [LR05] does not rely on these conditions. Instead, both approaches use another type of knowledge to generate workflows: in the case of [KWJ+04] this is biochemical knowledge; in the case of [LR05] it is historic performance data. Finally, both the Mealy FSM approach of Berardi [BGL+04] and the composition rule based approach of Medjahed [MBE03] enable to check for matching conditions for complementarity between parameters and operations. Although neither paper explicitly mentions supporting workflow fragment complementarity, their chosen technical

	[SVK+07]	[CGB06]	[KBL+07]	[MW06]
Complement based	yes	no	no	no
ComplPar	yes	no	no	no
ComplOp	yes	no	no	no
ComplW	yes	no	no	no
Identity based	yes	yes	yes	yes
SamePar	yes	yes	no	no
SameOp	yes	yes	no	yes
SameW	yes	yes	yes	yes

Table 3.6: Classification of workflow discovery tools against workflow matching conditions.

	[KWJ+04]	[SVK+07]	[BGL+04]	[LR05]	[MBE03]
Complement based	no	yes	yes	no	yes
ComplPar	no	yes	yes	no	yes
ComplOp	no	yes	yes	no	yes
ComplW	no	yes	yes	no	yes
Identity based	no	yes	no	no	no
SamePar	no	yes	no	no	no
SameOp	no	yes	no	no	no
SameW	no	yes	no	no	no

Table 3.7: Classification of workflow composition tools against workflow matching conditions.

approach makes it plausible that they do.

3.7 Summary and discussion

We defined workflow discovery, examined the role of discovery in scientific workflow re-use, specified technical requirements and considered multiple application scenarios. We scoped the problem to discovery based on the structural properties of a workflow.

We distinguished between discovery that is based on similarity, and discovery that is based on complementarity. Both types support different workflow re-use and repurposing tasks. We used the distinction to define workflow discovery tasks formally. In addition, the distinction was useful to classify existing approaches to workflow discovery.

We also developed a range of fine-grained matching conditions, detailing the degree of match between workflows at different levels. The conditions provide a rich framework against which to contrast discovery techniques.

- At a high level, they have proven useful to compare existing discovery techniques (particularly so when these involve data flows). None of the existing work is able to match all of the fine-grained conditions. The conditions will be re-used at a high level to compare techniques developed for the Taverna workbench in Chapter 5. Similarly, none of the techniques there match all conditions.
- At a detailed level, in future the matching conditions may provide a useful specification to design similarity and complementarity metrics against.

The next chapter will refine the requirements for workflow discovery by investigating how scientists, and in articular bioinformaticians, approach re-use and discovery.

Chapter 4

Building benchmarks for workflow re-use

This chapter presents the results of a series of user experiments. The goal of the experiments is to understand how bioinformaticians discover concrete workflows. The experiments are confined to the case where discovery supports the re-use task of adapting an existing workflow a user works on.

All five experiments use a corpus of real-world bioinformatics workflows, as generated by domain experts with the Taverna workbench.

4.1 Overview of experiments

The experiments differ widely in their setup, reflecting the different approaches taken for capturing how re-use occurs, the different conditions under which re-use occurs and practical constraints involved in running user experiments.

We use the term "experiment" because in each there is the underlying hypothesis that, under a fixed set of conditions, scientists can successfully perform workflow re-use and discovery. At the same time, the experiments are attempts at building benchmarks, where the successful outcome of the experiment yields a benchmark as a side effect.

Table 4.1 provides an overview of the experiments according to their wide differences in experimental setup, number of participants, materials, procedure and results. We discuss the details below.

	1	2	3	4	5
Setup					
Design	white box	black box	black box	black box	grey box
Re-use subtasks	discovery	discovery, editing	discovery	discovery	discovery, editing
Discovery task	similarity	sub, super	sub, super, alternative, equivalent	sub, super, alternative, equivalent	sub, super
Re-use directions	B2B, B2C	B2C	A2A	B2A	B2A, A2B, B2C
Participants					
Number	9	15	2	2	
Expertise	+	++	+++	++++	+++
Materials					
Use of survey	yes	no	no	no	yes
Workflows	6	20	67+78	33	24
Documentation	+	+	++	++	++++
Procedure	embedded	embedded	independent	independent	independent
Results					
Avg. duration	30 min.	20 min.	105 min.	30 min.	90 min.
Stat. support	-	-	++	+	++
Assessments	N/A	N/A	145	456	1848

Table 4.1: Overview of user experiments.

4.1.1 Experimental setup

The experimental designs we set up to capture re-use behaviour either regard a user's behaviour as a black box, capturing only the outcome of that behaviour, or as a white box, with a view to also capture the behaviour exhibited to reach that outcome. In a first experiment, described in Section 4.2, we pursue a white box approach to model how users re-use workflows. The negative results encountered lead us to consider a less ambitious black box approach in the next three experiments. The positive results from those in turn drive us to investigate a "grey box" approach, capturing more than the black box but less than the first white box based experiment.

Following the discussion on the difference between re-use and repurposing in the previous chapter, we consider both the discovery step and the editing (integration) step. Table 4.2 shows the type of information captured in each case, split over re-use subtask. All experiments aim at least to collect information covered by the black box approach, as shown in Table 4.1. The task of discovering relevant workflows from a

	During discovery	During editing
Black box	User-selected workflows	User-made edit actions
Grey box	User-selected workflows + type and amount of information used	User-made edit actions + type and amount of information used
White box	User-selected workflows + type and amount of information used + search strategies	(not studied)

Table 4.2: Type of information measured during the black, grey and white box based experiments.

repository is the subject of benchmarks 1-5 while the subsequent task of integrating a found workflow with the current workflow a user is editing is part of benchmarks 2 and 5.

The discovery step is investigated along different dimensions, presenting users with different tasks. In the first experiment, an assessment of similarity between workflows is measured, whilst the subsequent experiments focus more on identifying specific task relationships between workflows.

We take into account that workflows are re-usable between several parties in several directions. Figure 4.1 summarizes the five possible ways the author of a set of workflows "A" can re-use her own as well as other people's workflows. We discuss these below. Table 4.1 highlights which experiments investigate which re-use direction. It is important to make these distinctions because they are expected to influence the difficulty of a re-use task and the strategies people use to solve it, e.g. re-use of workflows one is familiar with should be easier. A distinction is made between *personal re-use*, involving only workflows authored by the re-user, and *cross-author re-use*, involving other workflows.

1. **Personal re-use: from A to A (A2A)** Re-use of workflows from a personal collection, which are either versions of the workflow one is currently working on, or previously built workflows with a different topic altogether (Figure 4.2).

2. **Cross author re-use: from B to A (B2A)** Re-use of workflows from someone else's collection to alter one's own current workflow (Figure 4.3).

3. **Cross author re-use: from A to B (A2B)** Re-use of workflows from one's own collection to alter a workflow from someone else's collection (Figure 4.4). The

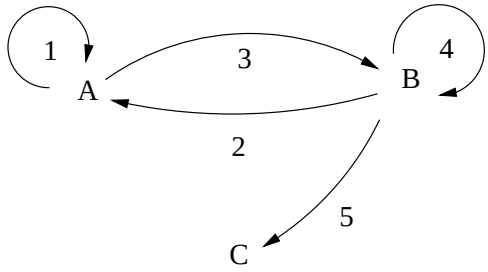


Figure 4.1: Types of workflow re-use from the perspective of the author of a set of workflows A.

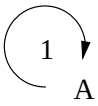


Figure 4.2: Single author re-use: from A to A (A2A).

difference between A2B and B2A is in the starting point – do we take our own workflow as a starting point and extend it with one of our own or do we extend our own with someone else's. Although the end result may be the same, the edit operations undertaken to integrate the two will be different.

4. **Cross author re-use: from B to B (B2B)** Re-use of workflows from someone else's collection to alter a workflow from that same collection (Figure 4.5).

5. **Cross author re-use: from B to C (B2C)** Re-use of workflows from someone else's collection to alter a workflow from another external collection (Figure 4.6).

4.1.2 Participants

Between two and 24 bioinformaticians participate in the conducted experiments. The expertise of participants ranges from MSc in Bioinformatics students novel to the idea

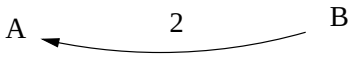


Figure 4.3: Cross author re-use: from B to A (B2A).

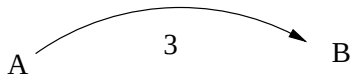


Figure 4.4: Cross author re-use: from A to B (A2B).

Figure 4.5: Cross author re-use: from B to B (B2B).

of workflows to experienced postdoctoral researchers having authored over 100 bioinformatics workflows. The expertise indication of participants assigned in Table 4.1 also appreciates that participants performing a re-use task based on workflows they authored are considered more expert than the other participants.

4.1.3 Materials

In two instances, surveys are used to gather feedback from participants before and after the experiments. In the other instances, informal interviews are used to probe for people's experiences.

To match the type of re-use task under investigation, different sets of workflows are used. The workflows are available publicly in the myExperiment.org repository.

The different experiments reveal different amounts of workflow detail to users. In its most basic form, a workflow as shown to a participant consists of an orchestration of services rendered as a diagram, showing only those inputs and outputs actively involved in the orchestration shown (+). A slightly more detailed version shows also the name of the workflow (++). In the case of the last experiment, even more detail is provided with the inclusion of textual descriptions of the overall workflow task and

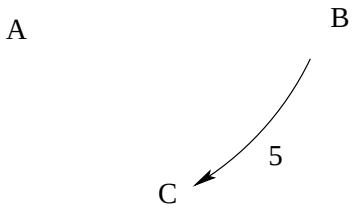


Figure 4.6: Cross author re-use: from B to C (B2C).

of the services as well as semantic annotation of services, based on concepts selected from the ^{my}Grid ontology (++++).

4.1.4 Procedure

Some experiments piggyback on already organized training days for the Taverna workflow editor. Others are set up independently through separate sessions.

4.1.5 Results

Based on the statistical analysis of their results, some experiments are more robust than others, resulting in assessments (not-) usable as a benchmark.

4.2 Experiment 1: cross author, white box re-use

4.2.1 Experimental setup

In this section, we pursue a white box approach to model how users re-use workflows across authors. The re-use directions captured are B2B and B2C. Specifically, we target the discovery of similar workflows and do not consider the editing of workflows found. The goal of the experiment is threefold:

1. Establish how bioinformaticians select relevant workflows based on providing similarity assessments. We use similarity as a broad metric to uncover relevant workflows for workflow re-use tasks. The intuition behind this approach is that workflows that are known to perform a subtask, supertask, alternative task or equivalent task of a given workflow could be considered similar to that workflow. Do participants think this task is hard? Do they behave consistently?

2. Gather feedback on the type and amount of workflow information users rely on for making (dis-)similarity assessments.

3. Uncover the patterns of searching and matching bioinformaticians use to establish workflow similarity. How do they rank workflows? Do people use some criteria all the time, regardless of the workflow in question? What is the relation between the matching criteria they use: for example, do they reinforce each other or cancel each other out?

We report on the data gathered during this user experiment and its statistical interpretation. Given the weak results of our statistical analysis afterwards, we are only able to provide a partial answer to the above questions.

4.2.2 Participants

During a ^{my}Grid User Day event (February 2006), 13 users completed an exercise to rate the similarity between an exemplar workflow (shown in Fig. 4.8) and other workflows. Nine users were bioinformaticians and four were software developers. None were an original author of the workflows in the corpus.

4.2.3 Materials

The user experiment included the use of an on-line survey, a corpus of similar on-line workflows and the rendition of those workflows. All are made available on-line.

An on-line survey

An on-line survey was created through the survey service at `www.keysurvey.com`. It contained three main sections. The first section gathered basic information on the subject and established whether they understood the biology behind the exemplar workflow. The second part asked users to rate the workflows and their confidence in doing so. The final part asked which additional information would be helpful in making similarity assessments and whether they found the exercise difficult.

A corpus of public workflows

A corpus of six workflows was used as the basis for the exercise. Five of the workflows in the corpus are created by Peter Li, a bioinformatician researching Graves disease [LHJ$^+$04]. Because the workflows are versions of each other, they are highly related and form a good basis for a workflow similarity experiment. The biological goal of this set of workflows is to discover genes involved in the disease based on microarray data and to prepare the genotyping of single nucleotide polymorphisms (SNPs) which are nucleotide variations that occur in those genes. The sixth workflow (see Fig. 4.13) is created by a different author, Hannah Tipney, to investigate the genetic basis of Williams Beuren syndrome [STW$^+$04]. The workflows can be accessed from within Taverna, as shown in Fig. 4.7. The exemplar workflow is shown in Fig. 4.8. The

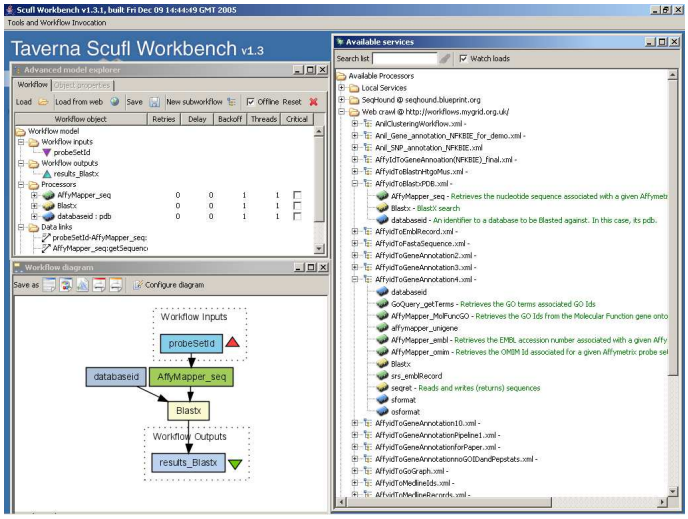


Figure 4.7: The AffyidToBlastxPDB.xml workflow loaded in the ^{my}Grid workbench. The Available services pane provides access to both services and workflows.

comparison workflows are shown in Figures 4.9 - 4.13. They differ on dimensions such as size, node orderings and differences in the scope of the biological task. The chosen combinations of workflows ensure that authors are tested on the B2B re-use direction for those combinations where only Peter Li's workflows are considered. The B2C direction is tested when the exemplar is used in conjunction with the Williams Beuren syndrome workflow.

4.2.4 Procedure

The survey was presented as an exercise that was an integral part of the training at the User Day. The stated goal of the exercise was to allow a user to study and try to understand some more complex workflow diagrams, while allowing the ^{my}Grid team to understand how similarity between workflows is perceived.

Users were first shown an overview of all available workflows, to give them an impression of the complexity involved in the manual discovery task. They were then explained the concept of a gold standard or benchmark. Five workflows from the corpus were presented for comparison against the exemplar workflow. For each workflow, users were presented with five questions to judge how similar it was to the exemplar workflow.

Participants were instructed to leave the question on similarity of biological functionality open when they did not understand the biology, and as a check we asked that

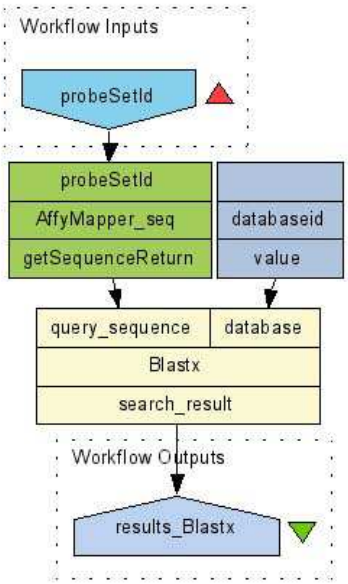


Figure 4.8: The exemplar workflow AffyidToBlastxPDB.xml in more detail. From a workflow input (accompanied by a red triangle), the workflow accesses the AffyMapper and BlastX Web services (the middle boxes) and yields an output (indicated by a green upside down triangle).

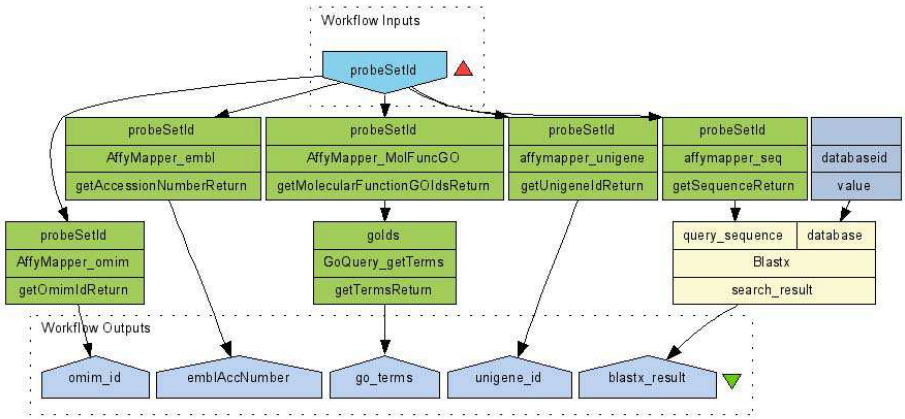


Figure 4.9: Workflow 1, AffyidToGeneAnnotation2.xml.

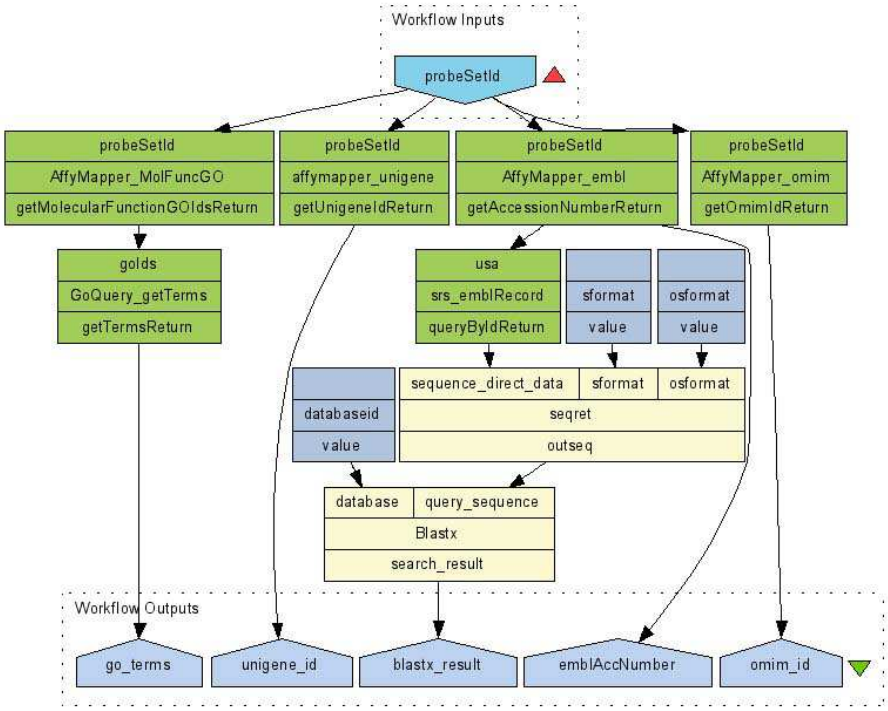


Figure 4.10: Workflow 2, AffyidToGeneAnnotation4.xml.

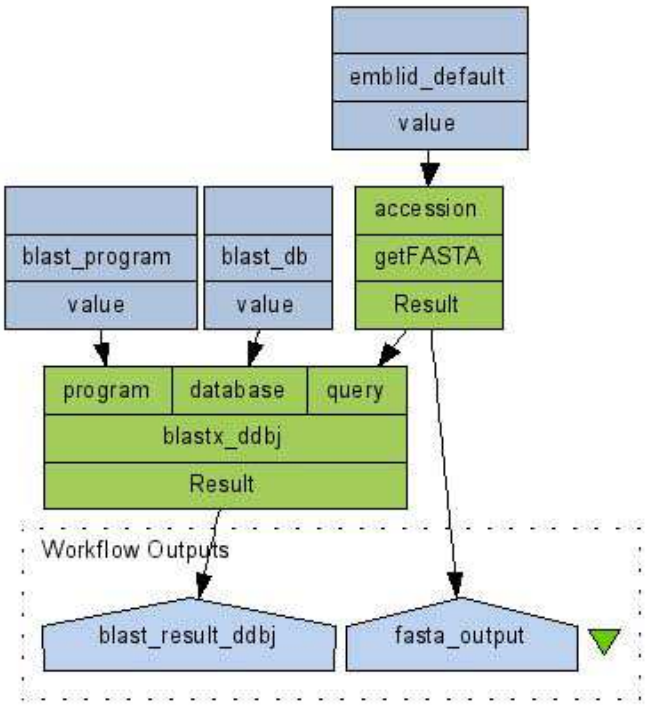


Figure 4.11: Workflow 3, BlastNagainstDDBJatDDBJ.xml.

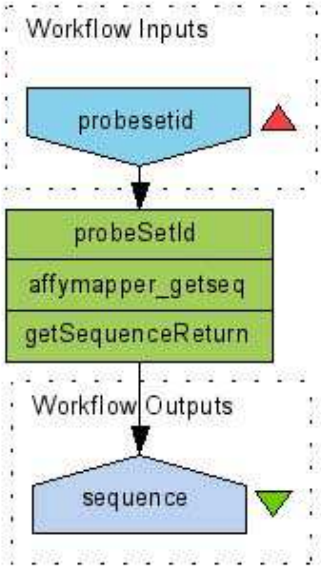


Figure 4.12: Workflow 4, AffyidToFastaSequence.xml.

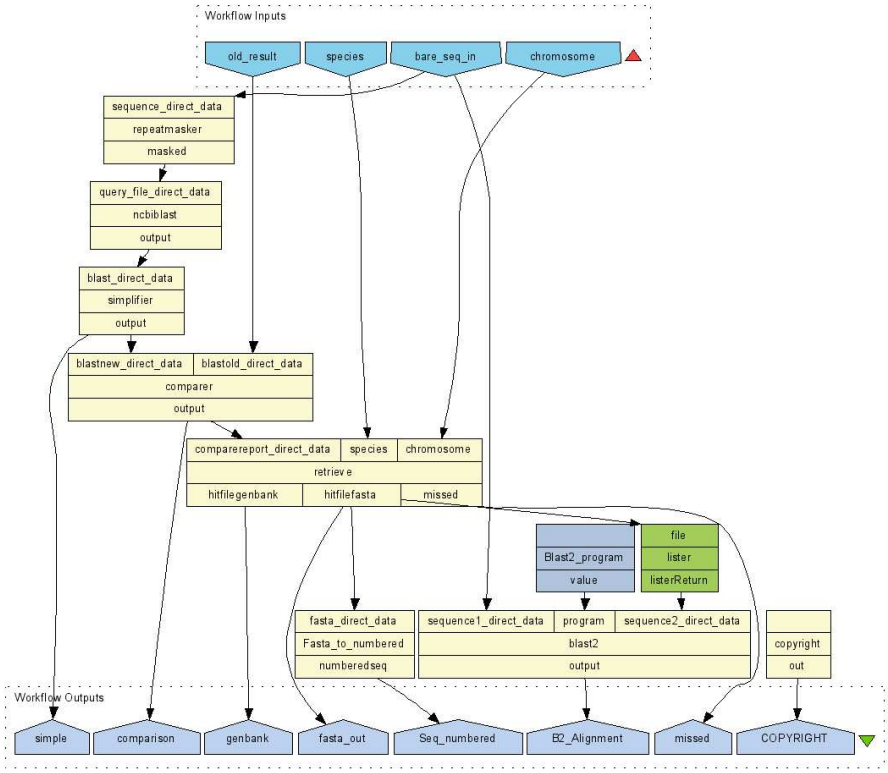


Figure 4.13: Workflow 5, williams-partA-paper.xml.

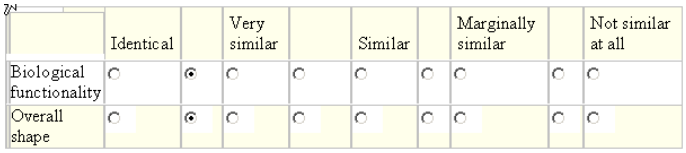


Figure 4.14: The form for entering workflow similarity values.

they describe the biology behind the exemplar and the comparison workflows.

To indicate the similarity between a pair of workflows, users selected a bullet from nine options (see Fig. 4.14). Each bullet corresponds to a value: 1 corresponds to `Identical`, 5 to `Similar`, and 9 to `Not similar at all`. Users also provided a measure of confidence in their similarity assessment, ranging over: `High--Medium--Low`, with `High` having value 1 and `Low` being equal to 5.

Finally, they had to rate how useful they found six factors for estimating similarity, with usefulness defined as: `Very useful-- Useful-- Only a bit useful-- Not useful at all`, with `Very useful` equal to 1 and `Not useful` equal to 4. The factors users were asked to rate were the following:

1. It makes biological sense to have this workflow as a part of the example workflow
2. It makes biological sense to have this workflow superimposing over the example workflow[1]
3. Workflow shape: number of shared inputs and outputs
4. Workflow shape: service type correspondence
5. Workflow shape: shared service compositions
6. Workflow shape: shared paths between (intermediary) input and output

4.2.5 Results

The entire exercise took participants 30 minutes on average. By analysing the generated data, we can partially answer the questions put forward in the beginning of the section. We used the SPSS (`www.spss.com`) statistical package for analysis. The full results are available on-line. The goals set are to establish which workflows participants select as more relevant; to uncover which type and amount of workflow information they use; and which common patterns they apply for ranking.

[1] By superimposing we mean one workflow performs a supertask of the other workflow.

No.	Biological similarity	Shape similarity	Confidence
1	4.5 (2.0)	5.0 (1.5)	3.0 (1.1)
2	5.8 (1.9)	6.8 (1.4)	2.9 (1.0)
3	7.0 (1.7)	6.8 (2.0)	3.3 (0.8)
4	6.2 (1.9)	5.0 (2.4)	2.9 (1.4)
5	8.0 (1.6)	8.4 (1.3)	3.2 (1.4)

Table 4.3: Similarity (1 = identical, 9 = no similarity) of 5 workflows with respect to the exemplar.

Agreement between participants

This section analyses whether the rankings users generated in the experiment are comparable. First, we determine what values were assigned on average to workflow pairs. Then, we establish whether there is a degree of consistency in these assessments, by calculating the degree of correlation in the answers given by different participants.

Similarity values assigned Users ranked the similarity of the exemplar workflow with respect to the comparison workflows as indicated numerically in Table 4.3.

The table shows the mean of similarity values given by all respondents per workflow. The standard deviation is given inside the brackets (two standard deviations away from the mean account for roughly 95 percent of the people).

The respondents reported medium confidence overall in their own judgment. Two thirds of respondents found estimating the similarity of biological functionality very difficult to difficult. The estimation of the similarity of shape similarity on the other hand was a difficult task for only 25 percent of respondents.

Participant agreement Did participants agree on their answers? To establish whether there was *consistency in the rankings*, one needs to know the degree of correlation between the different rankings. We performed the following calculations to obtain this statistic. The user similarity values of Table 4.3 were transformed for all participants to reflect the *order* in which each individual had ranked the workflows (in order of decreasing similarity). We then built a correlation matrix based on the transformed data and a statistical test called Spearman's correlation test. Spearman's correlation test is a measure of association between rank orders.[2]

[2]With respect to the data, for biological functionality similarity, we only used data from people with a biological background (nine respondents and five workflows, no missing values), whilst taking into account data from all 13 respondents for shape similarity.

No.	Sub task	Super task	I/O	Type	Services	I/O paths
1	2.4	2.0	2.5	2.2	2.1	2.0
	(1.0)	(0.9)	(0.9)	(0.8)	(0.5)	(0.6)
2	3.0	2.6	2.9	2.1	2.2	2.1
	(0.8)	(0.9)	(1.1)	(0.9)	(0.8)	(0.9)
3	3.0	3.3	2.6	2.2	2.5	2.3
	(0.8)	(0.8)	(0.8)	(0.8)	(0.5)	(0.7)
4	2.3	3.0	2.4	2.1	2.3	2.3
	(0.9)	(0.8)	(0.9)	(0.8)	(0.7)	(0.5)
5	3.6	3.3	2.5	2.5	2.5	2.5
	(0.7)	(1.2)	(1.1)	(1.1)	(1.1)	(1.1)

Table 4.4: Usefulness (1 = best, 4 = worst) of six factors for estimating similarity of 5 workflows with respect to the exemplar.

For rankings based on biological functionality, only for six out of 36 possible participant pairs (*e.g.* between participant 2 and participant 5) the results were correlated (at a five percent significance level). This means that participants in general disagreed on how to order the workflows according to their biological similarity and therefore lacked consistency in their rankings. A similar result was found in the correlation matrix for rankings based on shape: based on the 13 users, only 16 out of 77 pairs showed correlation (at a five percent significance level). As it stands, the current data set cannot serve as a general benchmark or gold standard.

Type and amount of information used and common patterns

One way to explain the diversity in behaviour during ranking we described above is to investigate whether different users use different criteria for establishing workflow (dis-) similarity. Users ranked the usefulness of the factors for establishing workflow similarity as indicated in Table 4.4. We are interested to find the effects of the six factors on the similarity, whilst the factors may be interacting (*e.g.* once people looked at how many services are shared, they might not care any more whether any inputs and outputs are shared). A common statistical way to establish such findings is through a Between-Subject Analysis of variance (ANOVA) (see `davidmlane.com/hyperstat` for a good introduction). Analysis of variance assumes that the groups come from populations with equal variances. To test this assumption, we used Levene's homogeneity-of-variance test, only to find that the assumption was violated in all cases. As a result,

little can be said on how particular factors impact the similarity measures based on ANOVA.

One explanation for the inconsistencies is that different people might be using different metrics, some of which are not included in the list of six factors. One participant for instance indicated that the total number of services (*i.e.* the difference in size of workflows) played a role in the assessment of shape.

With respect to future white box experiments, one logical solution would be to fix the (combination of) criteria people can use, and see what similarity values across participants this generates.

Impact on future experiments

Overall, the outcome of the experiment is rather negative. The overall message is that the task as presented to users is too difficult. This leads to inconsistent results. With respect to future experiments, this issue can be addressed by:

- Providing subjects with a clearer task definition, e.g. practical reuse tasks such as finding extensions to current workflows. We take this approach for benchmarks 2 and 5.
- Providing subjects with more information about the workflow, say to include textual and semantic annotation of the constituent services and overall workflow task. We invest in creating detailed workflow documentation for benchmark 5.
- Recruiting subjects with more experience with the workflow environment such as contributors to the myExperiment.org workflow portal. We take this approach for benchmarks 3,4 and 5.
- Recruiting subjects with more familiarity with the workflow corpus topic, for instance the original workflow authors. We take this approach for benchmarks 3,4 and 5.

4.3 Experiment 2: cross author, black box re-use

4.3.1 Experimental setup

In this section, we pursue a black box approach to model how users re-use workflows across authors. The re-use direction captured is B2C. Contrary to the previous experiment, we look at the discovery of relevant workflows in the context of a set re-use task. The task is to identify fragments in repository workflows that do either a supertask or a subtask of the provided exemplar workflow. The goal of the experiment is twofold:

1. Document which workflows bioinformaticians select as relevant to the set task through their decision whether or not a workflow is a match to an exemplar workflow.

2. Document which fragments of workflows bioinformaticians select as subtasks or supertasks to an exemplar workflow.

4.3.2 Participants

During a ^{my}Grid Training Day event (April 2006), five out of a total of 21 participants completed the exercise. Against expectations, the majority of participants of the training day were software developers or scientists with no bioinformatics background. The five eligible participants were full-time bioinformaticians without earlier experience with workflows.

4.3.3 Materials

We used a corpus of workflows and the rendition of those workflows. All are made available on-line.

A corpus of public workflows

We used the `Probeset_id_2_Swissport_id.xml` [3] workflow authored by Paul Fisher as the exemplar workflow. 29 workflows were pre-selected from Peter Li's corpus as the repository, to include known cases where a subtask, supertask or no relationship is present between the exemplar and the repository workflow. All workflows were rendered on a diagram showing the inputs and outputs of services used in the

[3] We chose not to correct the SWISSPROT spelling error for authenticity.

orchestration. They were printed on separate pages of A4 format and stapled together. The exemplar workflow was shown on top. Workflows diagrams were stripped from all colour since colour reflects a Taverna Processor's type (Web service, local Java class, ..), which does not matter for this exercise and it was a source of confusion in the previous experiment.

4.3.4 Procedure

The experiment was presented as an exercise that was an integral part of the Training Day. The stated goal of the exercise was to contribute to the ^{my}Grid team's understanding of workflow re-use. Users were handed out the stapled stacks of workflow diagrams and asked to remove the exemplar on top and use it alongside the stack while flipping through it. They were then asked to mark up the workflows as being a match or no match, and to circle relevant areas on the diagrams that were considered supertasks and subtasks.

4.3.5 Results

Participants took 20 minutes on average to do the workflow comparisons. Out of five participants, only one completed the full set of comparisons.

On the upside, the one participant that completed the exercise showed good correlation with our pre-selected sets of subtasks, supertasks and unrelated tasks. This suggests that re-use by bioinformaticians is in fact doable based on limited workflow documentation.

On the downside, the other participants were generally frustrated with the experiment and thought it was too hard. Their motivation to assess the suitability of workflow candidates was affected by several factors. Some diagrams proved to have a too small print, making it a chore to interpret them. Secondly, the lack of textual descriptions of what the services and workflows do proved a hurdle, and made people unwilling to investigate whether there exists a relationship between the workflows. Finally, the fact that software developers and the other scientists did not "have to" participate in the exercise did not help motivation.

When it comes to re-use, there is the trade-off between doing things yourself and making the effort to find out whether existing things can be re-used. If scientists novel to workflows are to make the effort of assessing their suitability for re-use, documentation beyond what is captured in the basic diagram is going to be needed.

Given our goal to capture how bioinformaticians do re-use (when they are motivated enough), in the next experiments, we address the concerns raised by resizing diagrams, creating workflow documentation, selecting participants with a higher level of expertise and organizing dedicated experiment sessions.

4.4 Experiment 3: cross author, black box re-use

4.4.1 Experimental setup

This experiment takes a black box approach to model how users re-use workflows across authors, focusing on re-use direction B2A. We look at the discovery of useful workflows in the context of a specific re-use task. The task for the participants is kept simple by asking for boolean assessments of "usefulness" instead of similarity scores and the metrics used during matching.

Useful is specified to mean that one workflow either (i) provides an alternative to the other, (ii) provides an extension to it (supertask) or (iii) provides a useful fragment of it (subtask). For example, each participant was asked to assess whether the pair of the `AffyidToGeneAnnotation2` workflow, taken from one corpus, and the `Probeset_id_ to_Swissport_id`, taken from another, were useful taken together, or not.

4.4.2 Participants and procedure

The results of the previous experiments lead us to reconsider the number of participants in favour of fewer participants but with more workflow experience and familiarity with the corpus.

Peter Li and Paul Fisher are the two bioinformaticians participating in the study. In what follows they are denoted as expert 1 and expert 2, respectively. Together they authored some 300 bioinformatics workflows with the Taverna workbench. They were asked to make an assessment of similarity between their own workflows and workflows created by the other expert. Both were contacted separately and asked to perform the exercise in our presence.

4.4.3 Materials

All data sets are available from `www.myexperiment.org/benchmarks`.

Workflow corpora

We chose a corpus that consists of 67 workflows investigating Graves' disease [LHJ+04] authored by expert 1 and 78 investigating Trypasonomiasis (sleeping sickness) in cattle, authored by expert 2. We randomly chose 19 workflows built by expert 1 and 11 workflows built by expert 2 from their respective corpora.[4]

4.4.4 Procedure

We made the random selection in the following way. We took the amount of time experts can reasonably spend on an exercise without getting tired (about two hours), and divided that by the average amount of time it took experts to make a few example comparisons (30 seconds). We added one dummy workflow to expert 2's set to be able to check whether the experts are internally consistent in their assessments. The two sets were printed and collected in two bundles of A4 paper.

Each participant was given the stacks of workflow diagrams and asked to assess whether pairs of workflows taken from the 2 different corpora were useful (as defined earlier) for each other, or not.

4.4.5 Results

The result of the exercise were two sets of assessments of usefulness, each 12 x 19 in size, *i.e.* a total of 456 observations. To see whether the experts were *internally* consistent, as said we added a dummy workflow. Both authors handled the appearance of the dummy well.

To establish whether there was consistency *between* the authors, we calculated the kappa coefficient for inter rater agreement, as commonly used in psychology experiments. Table 4.5 shows how often the experts (dis-)agreed on whether 2 workflows were useful for each other. This translates in a kappa measure of 0.678, which indicates a good level of agreement between the subjects. Table 4.5 shows the data used to calculate kappa. It details how often the experts (dis-)agreed on whether two workflows were useful. The numbers indicate how many times the experts agreed and disagreed with each other. For example, there are 11 cases where expert 1 thought a pair of workflows was useful together, but where expert 2 thought they were not.

Building the benchmark took Paul 2 hours, and Peter 1.5 hours. We merged the

[4]There is no particular reason why more workflows were chosen from expert 1 than from expert 2.

	Expert 2 not useful	Expert 2 useful	Total
Expert 1 not useful	148	19	167
Expert 1 useful	11	50	61
Total	159	69	228

Table 4.5: Agreement between the experts in their workflow exercise assessments.

results of this exercise to form 1 benchmark, with a bias towards positive similarity assessments, *i.e.* in case only one of the experts thought a pair was similar, we still included it in the benchmark as similar.

4.5 Experiment 4: personal, black box re-use

4.5.1 Experimental setup

This experiment is similar to the previous one in its setup. It again takes a black box approach to model how users re-use workflows, based on participants with a high level of expertise and familiarity with the corpus. This time we consider personal re-use, however (re-use direction A2A). The task set for participants is to identify different versions of a given workflow authored by the participant, where the retained workflows include both the most recent and earlier versions of a given exemplar workflow.

4.5.2 Participants and procedure

Peter Li and Paul Fisher are again the two bioinformaticians participating in the study, denoted as expert 1 and expert 2, respectively. They were contacted separately and asked to perform the exercise while we were present.

4.5.3 Materials

All data sets are available are available from the ^{my}Experiment Wiki.[5].

[5]Web site: `www.myexperiment.org/benchmarks`

Workflow corpora

We used 67 workflows built by expert 1 and 78 built by expert 2. For both the expert 1 and expert 2 subcorpus we chose a workflow which clearly had a lot of relation to the work the authors had done in the past. We asked expert 1 to identify different versions of his `AffyidToGeneAnnotation2` workflow in his corpus and asked expert 2 to do the same for his corpus based on his `Probeset_id_ to_Swissport_id` workflow.

4.5.4 Results

Each expert built a benchmark based on their respective corpus. Versioning benchmark 1 consisted of all 48 workflows identified by expert 1 as different versions of `AffyidToGeneAnnotation2` based on his corpus. Versioning benchmark 2 created by expert 2 identified 19 different versions of `Probeset_id_to_Swissport_id` within his workflows. Building the benchmark took each author about half an hour.

4.6 Experiment 5: cross author, grey box re-use

4.6.1 Experimental setup

Experiments 1-4 explored either a white box or black box approach to modelling user behaviour during discovery and editing time. They established that it is difficult to obtain sound results for a white box approach and that it is possible to obtain statistically valid results for a black box approach. This experiment explores the middle ground or "grey box" in that it aims to:

1. elicit the steps participants undertake to go about solving a re-use problem in a controlled environment;

2. elicit which types of workflow information matter during which steps; and

3. capture the resulting solutions in a fine-grained manner by documenting the particular workflow edit operations undertaken.

In terms of the re-use subtasks undertaken, the experiment is set up to capture both workflow discovery and workflow editing. In terms of workflow edit operations undertaken, workflow insertion, replacement and extension are investigated. In terms

of the re-use directions, we look at cross-author discovery (in particular, the B2A, A2B and B2C scenarios).

4.6.2 Participants

We ensured a high number of participants with a high level of expertise. Participants were either seasoned Taverna users or experienced bioinformaticians. The list of 24 participants is given in Appendix A.

4.6.3 Materials

Workflow re-use exercises were developed in conjunction with Franck Tanoh, service curator of the ^{my}Grid project,[6] and Paul Fisher, bioinformatician. 18 workflows from 12 authors were selected to ensure a broad topical range and variety in workflow authoring naming style. All data is available from the ^{my}Experiment Wiki at `www.myexperiment.org/benchmarks`.

Workflow annotation

The workflow documentation was of high quality. All constituting services were tested and then curated with natural language and ontology terms by Franck Tanoh, the ^{my}Grid service curator. For two months he semantically annotated 18 workflows containing 98 Web services. One of the 18 curated workflows is shown in Fig. 4.18. Next to the workflow diagram, it shows the workflow's overall task, overall inputs and outputs and constituting service inputs and outputs. In addition, semantic tags from the ^{my}Grid bioinformatics service ontology were added.[7] Workflows were annotated in two stages:[8]

1) Black box annotation. The entire workflow is regarded as a Web service with inputs and output parameters. A general description of what the workflow does is added. Associated ontological terms to the description are added between brackets, e.g. [retrieving], [pairwise_local_alignment].

2) White box annotation. This comprises a detailed annotation of each Web service in the workflow. It provides the following information on each service:

[6] Web site: http://www.mygrid.org.uk

[7] The ontology is navigateable at `http://www.mygrid.org.uk/ontology/OwlDoc/index.html`

[8] The following description of the annotation process is due to Franck Tanoh.

- Service name.
- Name of the service provider.
- Service type e.g. WSDL, Soaplab.
- Service description and ontological terms associated to it. These terms are defined as, operation Task, operation Resource and operation Method.
- Each input and output parameter and its semantic type and format.

Figure 4.15 describes how the annotation is carried out.

1. The first and very difficult step is to find the associated documentation of a given workflow or Web service.

2. The documentation is then carefully interpreted and the workflow tested using Taverna.

3. Information gathered from the documentation, the test, and the domain ontology is entered in an annotation tool resulting in semantically annotated workflows and Web services. At this stage, if missing terms are encountered in the ontology, ontology experts are notified.

4. Annotated workflows and Web services are published to a local registry where they will be used and tested by bioinformaticians and biologists.

5. Based on user feedback more description text or ontological terms can be added to previously annotated workflows and Web services.

To date, most of the Taverna workflows do not have natural language documentation by default. This lack of documentation greatly affects the speed of annotation. In some cases, workflow providers need to be contacted in order to annotate their workflow. As the number of workflows and Web services grows, workflow and service providers will need to follow suit and annotate their own workflows or services.

Exercise format

Given the lack of capabilities for editing multiple workflows in existing workflow systems, we decided to adopt a paper solution for the purposes of the experiment. For the workflows in our selected corpus, the amount of workflow information divulged meant a minimum of one A4 sized piece of paper was needed per workflow and a maximum of four A4's.

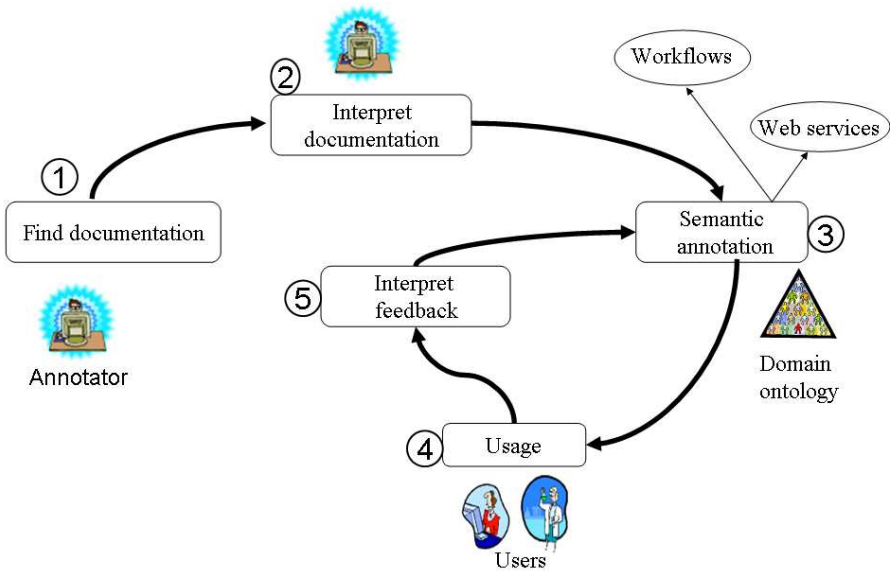


Figure 4.15: The workflow annotation process.

Given our aim to document both how participants discover workflows and then how they go about editing them subsequently, we wanted to make sure participants were able to keep the overview in the process. As a result, we adopted A1 poster sheets to lay out the re-use task description, the workflow to be adapted in the middle of the poster and the repository workflows huddled around it.

Fig. 4.16 shows the solved example exercise shown to participants, shrunk from its original A1 sized format. The challenge is to to solve the task stated at the bottom by adapting the workflow with the circled number (number 1, shown in detail in Fig. 4.18) using drawings on the poster. The particular task in question is to discover and then edit the workflow(-s) which enable the given workflow to obtain a list of gene identifiers by simplifying its BLAST output file. Note that we do not inform participants which edit operation to undertake (in this case, an extension).

To make clear to participants which possibilities they have at their disposal for editing workflows, the instructions directed them to Fig. 4.17 (again shrunk from its original A1 sized format).

4.6.4 Procedure

We briefly describe the pilot studies we ran, the distribution method and the analysis method.

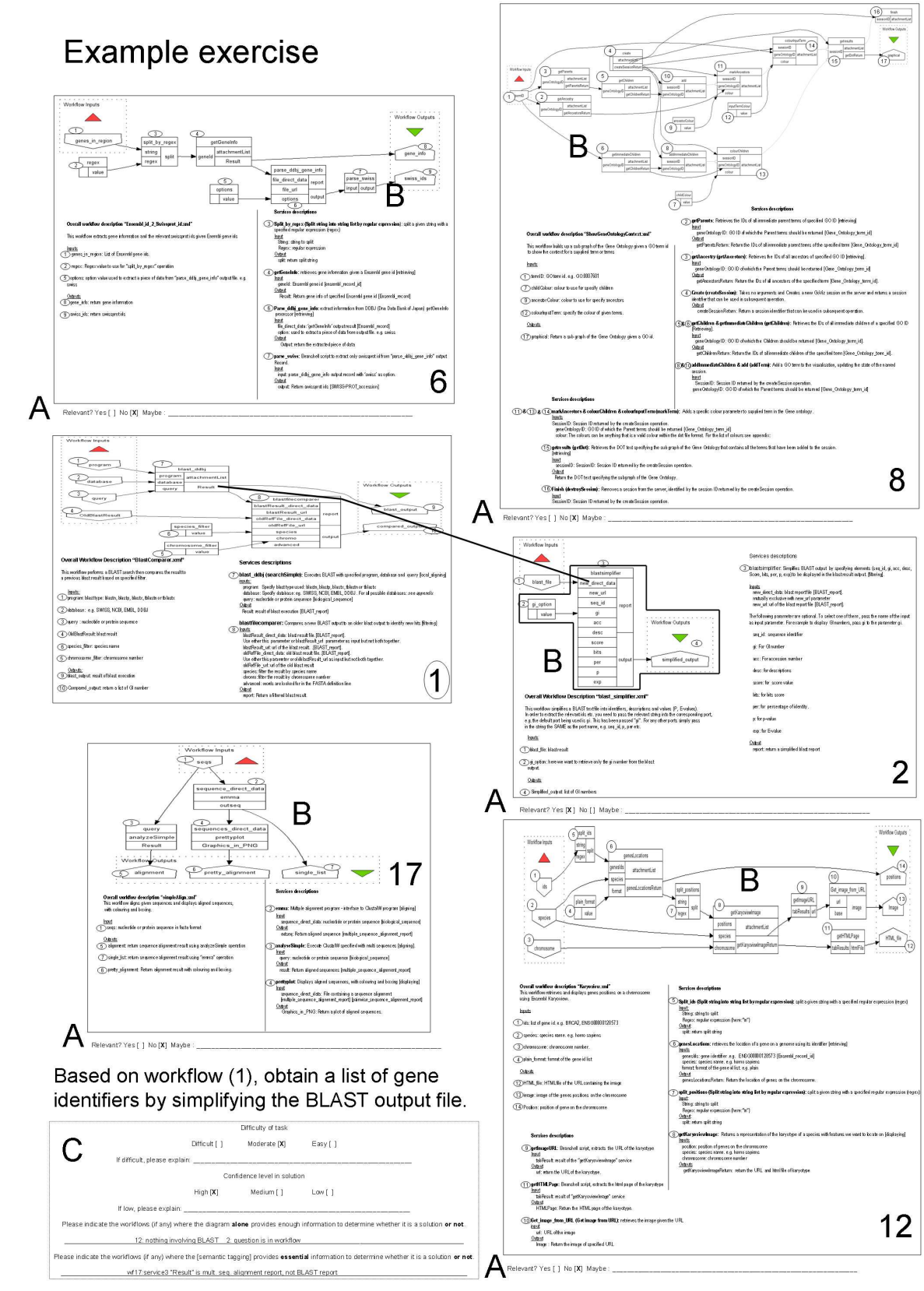


Figure 4.16: An example workflow exercise in user experiment 5.

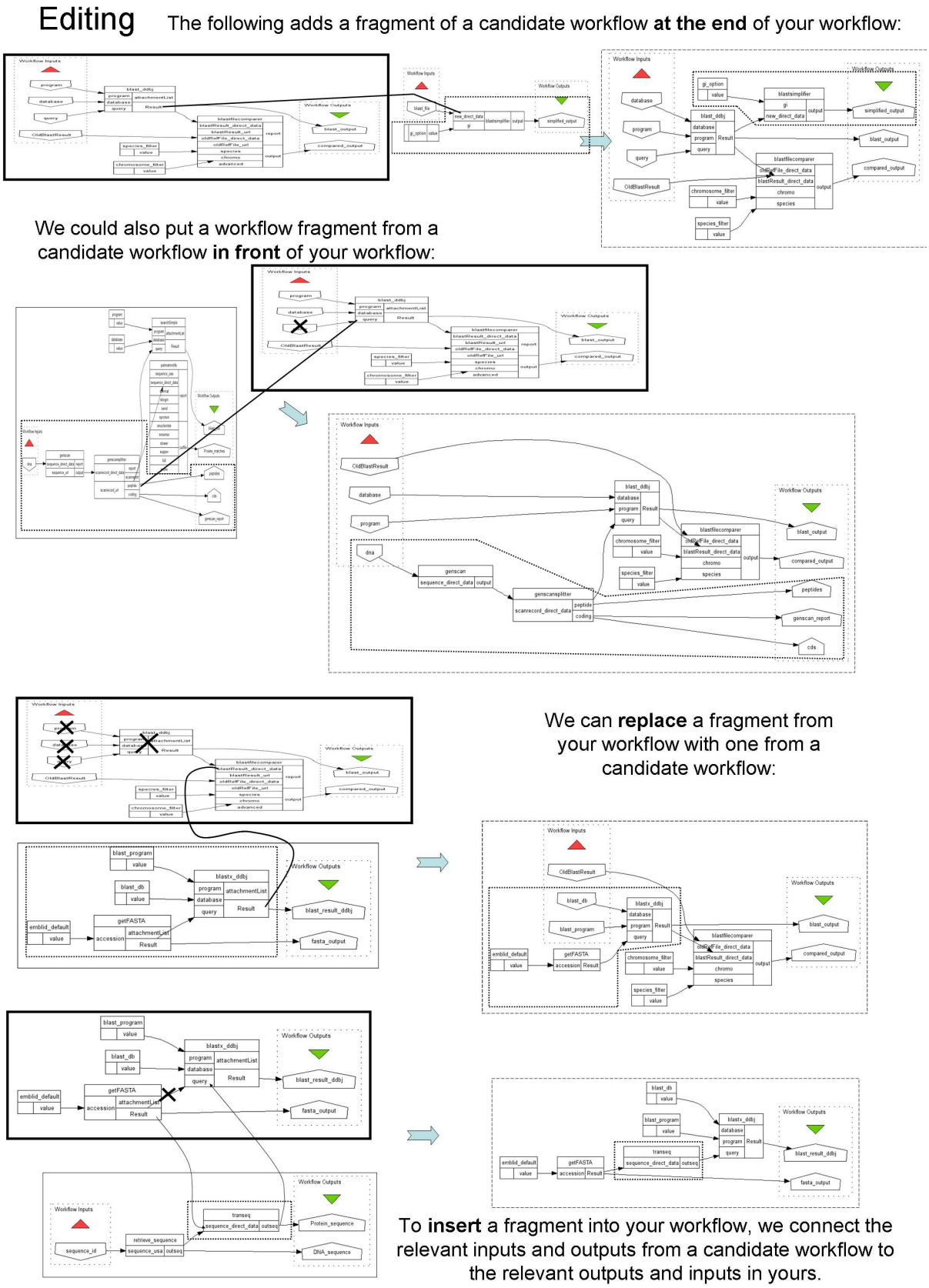


Figure 4.17: Workflow edit operations allowed by participants in user experiment 5.

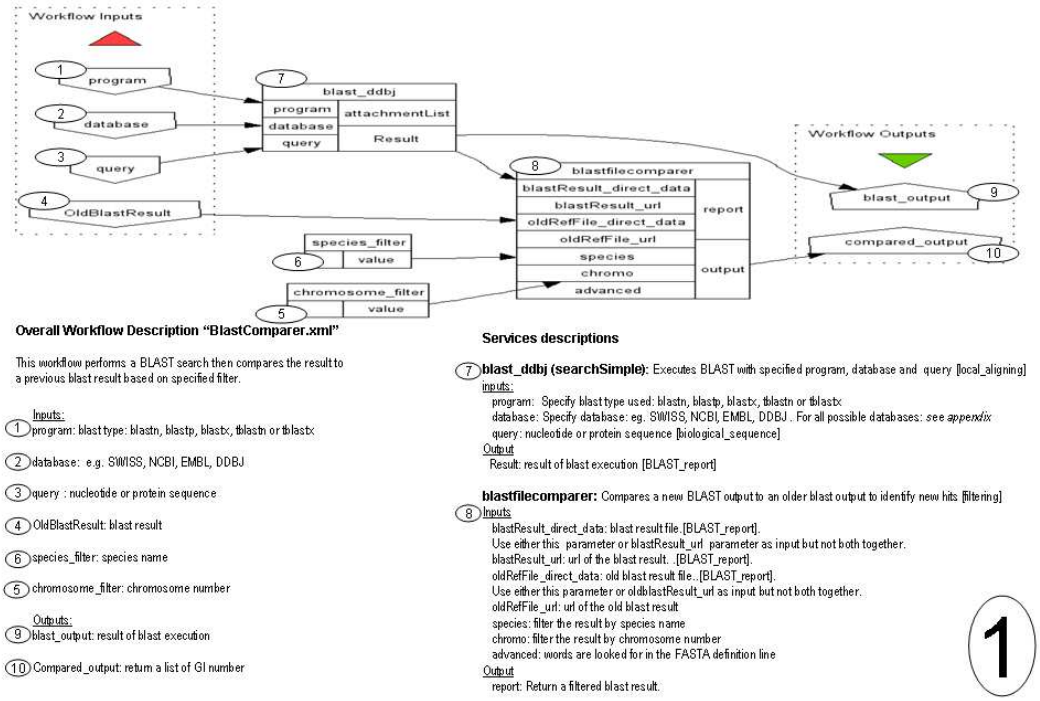


Figure 4.18: An example curated workflow

Pilot studies

The exercises were tested in two pilot studies with two post-doctoral bioinformaticians, Katy Wolstencroft and Andrew Gibson. Their feedback led us to adapt the vocabulary used in the exercise instructions for the target audience, to refine the curation and to remove ambiguities in the re-use task descriptions. Paul Fisher verified the validity of the correct solutions to a given task by creating the corresponding workflows in Taverna for all tasks.

Distribution

Remote participants were sent a tube by Royal Mail containing the instructions, the A1 posters and some quality Belgian chocolates. To local participants a tube was hand-delivered.

Analysis method

Results of the participants on the 11 exercises were entered into SPSS based on the following seven variables: Difficulty, Confidence, Relevance of (up to 5) candidate workflows. They were analysed based on Spearman correlation and Kappa values for inter-rater agreement. Given that Cohen's traditional Kappa value works for only two participants, we relied on a multi-rater Kappa measure, described in [SC88] and made available for SPSS as a separate macro.[9] Performance results against the correct answers were calculated from Excel.

4.6.5 Results

33 tubes were distributed and 24 were returned. Based on the above seven variables, 1848 observations were entered into SPSS. The edit operations made by participants were not processed due to time constraints.

Analysis of ratings shows participants in general had high confidence (Figure 4.20; blank in the legend means no answer provided) and found the exercises to be of easy to moderate difficulty (Figure 4.19; blank in the legend means no answer provided). Surprisingly, analysis of inter-rater agreement showed that they did not agree which exercises were easy, moderate or difficult. Similarly, they did not agree when they had

[9] Available from `ftp://ftp.spss.com/pub/spss/statistics/nichols/macros/mkappasc.txt`

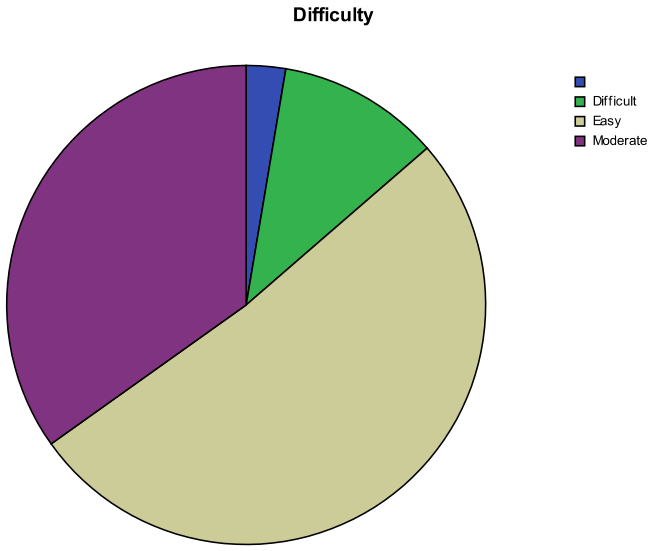


Figure 4.19: Perceived difficulty of the exercises during experiment 5.

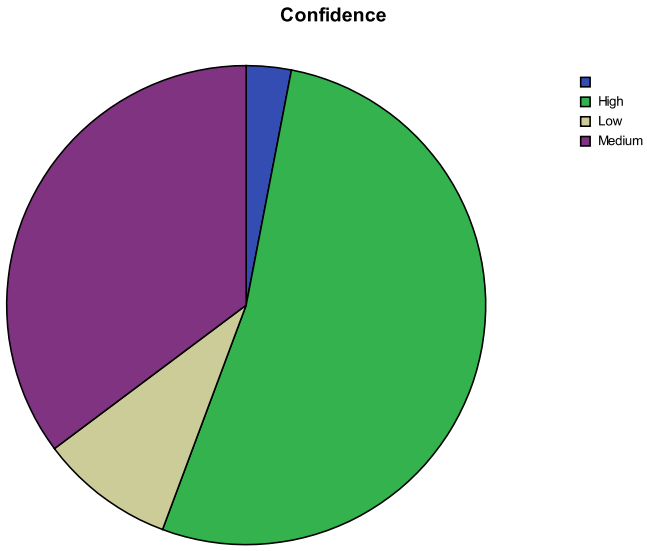


Figure 4.20: Participant confidence during experiment 5.

	Blanks excluded	Blanks mean no
Maybe excluded	79	83
Maybe included	87	91

Table 4.6: Average scores of participants by assessment scheme.

high, medium or low confidence. An explanation for this apparent paradox is either that participants come from very different backgrounds and thus find different tasks challenging, or they use a different internal scale to assess confidence and difficulty. Their results on relevance assessments suggest the latter is true.

Participants agreed well on the relevance assessments made - a multi-rater Kappa value of 0.666 was obtained, which is considered to be very good (for a discussion of the multi-rater Kappa measure see [SC88]). Contrasting participant relevance assessments with the correct solution, participants on average were right in 83% to 91% of all cases, depending on the scheme used to assess a given answer. Table 4.6 summarises their performance (in percent) based on four assessment schemes. The schemes vary on:

- whether they should count a "maybe" answer as a correct answer (which leads to better scores) or whether it should be excluded from the performance measure.
- whether blank answers should count as negative answers (which leads to better scores, given that the majority of candidate workflows are not relevant to a particular task) or instead should be excluded.

The main sources of error were due to:

1. Incomplete exercises because of a natural "*blind spot*" for participants in the exercise material, namely the workflow positioned in the left down corner of the A1 sheets. This phenomenon had not shown up in the pilot studies.
2. Incomplete or ambiguous *descriptions of data items*.
3. Assumptions made on the required *generality of a solution* across species.
4. Assumptions made on the *admissibility of additional "shim" or glue services*, not available from any of the presented workflows.

Finally, we analysed whether the amount of expertise building workflows or the time taken to complete the exercise showed a correlation with the level of correctness

obtained. Neither factors proved to be determinants. This indicates that people with a good bioinformatics background in general can muster the tasks of editing workflow diagrams and that some people simply work faster than others.

4.7 Related work

In this chapter we built a series of human benchmarks for workflow discovery. A a large number of possibilities was explored, ranging from eliciting the metrics experts use to establish similarity to capturing solely the end result of a task assessment in a given context, i.e. the collection of workflows an expert came up with to solve a task. Neither the workflow literature nor the Web services literature contain many examples of such work. Other disciplines have more of a tradition in using human-generated benchmarks to assess the performance of automated techniques, e.g. computational linguistics.

In the *workflow literature*, not much work exists on building human benchmarks. Recent work in the area has aimed to uncover the particular metrics people use for establishing workflow similarity. Bernstein and colleagues [BKBK05] look for the best semantic similarity measures to rank business processes from the MIT Process Handbook, based on a process ontology. The processes are non-executable workflows hence no reuse of workflows in a Web services context is envisioned. The work of Wombacher [Wom06][WR06] seeks to elicit the similarity metrics used by workflow researchers when performing the task of comparing the control flow complexity of workflows described by Finite State Machines (FSMs). Data flow is left outside the scope. Wombacher also investigates which metrics, known from workflow mining and FSM techniques, are able to reproduce the human rankings from this task.

In the *service discovery literature*, most of the papers presenting techniques ignore how humans go about discovery and focus instead on a technical evaluation, demonstrating how expressive a technique is, or how scalable. An exception is the work by Dong [DHM+04], who built a small human-built benchmark based on real Web services to test the performance of the Woogle tool for Web service search. We know of two community initiatives to compare Web service discovery techniques: the Semantic Web Services Challenge and the Web Service Challenge[10]. Both initiatives have limited involvement from users. In the former, a challenging scenario is put forward

[10]Web sites: `ws-challenge.org` and `sws-challenge.org`

involving fully automated discovery and invocation. In the latter, techniques are evaluated by a subjective score issued by the organizers on the system design as well as on performance and accuracy.

Other fields have been more active in developing human benchmarks for evaluating computational techniques. The practice is commonplace in for example Natural Language Generation (see for instance [Kar03]).

4.8 Summary

The apparent lack of end user-based evaluation for discovery tools is an important finding. It motivates the development of benchmarks for workflow discovery and the assessment of workflow techniques on these benchmarks.

In the controlled environment of a series of user experiments, this chapter explored how one class of scientists, bioinformaticians, do workflow re-use and discovery. The following findings were obtained.

4.8.1 Workflow re-use and discovery requirements confirmed

The *relative impact of several of the earlier posited requirements on re-use and discovery* were tested. Table 4.7 summarises the outcomes. We altered the following parameters between the experiments:

- *The discovery task expected of participants.* The discovery task involved either finding similar workflows or complementary workflows. In the last experiment, we also recorded how participants edited the workflows found.
- *The relation of the participant with respect to workflow authorship.* She is either discovering her own workflow, re-using one by a collaborator or one by an external party.
- *Participant motivation to complete the exercises.* This ranged ranged from low to high in the experiments, depending on the experimental setup.
- *The expertise of participants*. Low expertise means participants had almost no experience with bioinformatics. Medium means their main background is in another field but they understand basic bioinformatics notions. High means they are active in bioinformatics research.
- The *quality and amount of workflow information* available per workflow. Low quality and amount means only a workflow's diagram is shown with the names

Exp.	Discovery task	Relationship with original workflow author	Motivation	Expertise	Documentation quality and amount	Outcome
1	Find similar	External parties	Low	Low	Low	Negative
2	Find similar	External parties	Low	Medium	Low	Negative
3	Find similar	Original author	High	High	Low	Positive
4	Find similar	Collaborators	High	High	Low	Positive
5	Find and edit similar and complementary	Collaborators and external parties	High	High	High	Positive

Table 4.7: Summary of user experiments into workflow discovery.

given by the original author. High quality and amount means a professional curator added natural language text explanations and semantic concepts to the workflow and its constituent services.

The outcome of each experiment was judged to be positive only when the results from the exercises showed a level of agreement between participants and were confirmed by a bioinformatician as being sensible. The experiments revealed the following:

- *Bioinformaticians are capable of all types of workflow discovery* we considered, when the conditions are right.
- The commonsense expectation is that participant familiarity with the workflow author, participant motivation and participant expertise correlate positively with valid answers to discovery tasks. This expectation was confirmed in all experiments.

- Lots of quality workflow documentation is no requirement to achieve good results when it comes to discovery of one's own workflows or workflows by collaborators, as shown by experiments 3 and 4 which had low amounts of documentation (see Section 4.1.3).

- Experiment 5 showed that the combination of motivation, expertise and quality metadata enables discovery from external parties. Contrasting this finding with the outcome of experiment 2 where participant expertise was medium suggests that either the motivation factor or the quality documentation factor could be the deciding one. Taking into account that the motivation in experiments 1 and 2 was affected substantially by the lack of documention, we concluded that *documentation plays a crucial role* either indirectly (to drive motivation) or directly (to inform the discovery process).

4.8.2 Understanding of workflow re-use and discovery behaviour

Experiments 1 and 5 were designed to create an understanding of the behaviour bioinformaticians exhibit during workflow re-use and discovery. Experiment 1 formulated *a range of plausible hypotheses* to uncover patterns of searching and matching bioinformaticians use to establish workflow similarity in general. How do they rank workflows? What criteria are used and in which combination? However, it turned out the task as presented to inexperienced bioinformaticians was too difficult and no conclusive results could be drawn.

Experiment 5 also had the ambition to model bioinformatician behaviour, yet focussed less on the notion of the similarity and more on *how the overall re-use process is performed*, in particular the *assessment of relevance of potential workflow candidate solutions and their subsequent editing*. It presented specific re-use cases to experienced bioinformaticians.

The same study showed that *relevance assessment and editing are done in two distinct phases*. First, participants scanned the whole population of available workflows. After this, editing was done on the workflows marked as relevant.

It also documented *which sources of information are used in which phase*. For both phases, the workflow diagram was the first and most used point of recourse for finding information, despite its low detail and ambiguity. This finding underlines the power of using a visual medium. Textual workflow and service inputs and outputs were also used eagerly in both phases, but less so than the diagram. The overall workflow

Benchmark	Experiment	Number of participants	Behaviour captured	Number of assessments	Participant agreement (Kappa value)
1	3	2	Similarity assessments	145	N/A
2	4	2	Similarity assessments	456	Very good (0.678)
3	5	24	Relevance assessments	1848	Very good (0.666)
4	5	24	Editing	Unprocessed	Unprocessed

Table 4.8: Summary of human benchmarks for workflow re-use

description and workflow name were deemed useful for relevance assessment only.

The work of participants translates into a documented set of decisions made during the workflow re-use process. The three user experiments which had positive outcomes contribute four benchmarks with different characteristics.

Benchmark 1 collects similarity assessments made by a workflow author about pairs of her own workflows. In Benchmark 2, a collaborator made similarity assessments on those same workflows. Benchmarks 3 and 4 rely on the re-use tasks solved by participants in user experiment 5. Benchmark 3 contains the assessments made regarding the relevance of a set of workflows to solve the re-use task in question. Benchmark 4 captures the next step in solving the task by collecting the edit operations participants undertook to solve it. Due to time constraints, these edit operations have not been entered into electronic format yet and their statistical analysis has not been performed.

All benchmarks were created by participants who felt confident while creating them. For benchmarks 2 and 3, they also agreed strongly on the assessments made, as shown by the Kappa statistic for inter-rater agreement. Even though a high level of agreement was reached, agreement was never perfect. For the case of benchmark 3, the disagreement could be measured in terms of correctness of answers.

Chapter 5

Workflow discovery techniques

The user experiments showed that workflow re-use and discovery is difficult for bioinformaticians. To support them, in this chapter we consider a range of automated discovery techniques, tailored to workflows created in the Taverna workflow environment.

5.1 Overview of techniques

All the techniques investigated in this chapter work based on similarity-based matching of workflows. None were designed explicitly to support complementarity-based matching between workflows. The techniques differ mainly on the type of information they exploit. In this introductory section we discuss the nature of Taverna workflows, list the various sources of documentation used by the techniques and how the techniques relate to existing work. We also provide background to the structure used to organise the techniques in this chapter.

5.1.1 Data flows in Taverna

All techniques in this chapter are designed to work over workflows written with the Taverna workbench [OGA+05]. Taverna was chosen given its large user base of scientists and its publicly available workflow base at `www.myexperiment.org`.

Taverna has adopted a formal semantics based on data flow. It implements a lambda calculus with a list monad [Tur06]. In terms of the formal definition of a data flow adopted in Chapter 2, Taverna workflows deviate from our assumptions as follows:

1. Not only concrete workflows. In Taverna 2, it is possible to dynamically select at run time which service to run from a pre-specified group of services, violating

our assumption that all services which will be used are known at design time.

2. Not only Web services. Taverna steers multiple service types.

The techniques presented here are only able to deal with concrete workflows. In practice, this is not problematic for our purposes, given that all workflows in our corpora are concrete workflows. The second issue is more worrisome. It lead us to work at the level of abstraction that is specific to Taverna, namely that of the `Processor` [OGA+05]. With the exception of those techniques that work solely over semantic descriptions, our techniques are specific to the Taverna environment.

5.1.2 Source of workflow documention

Given the expense of obtaining high quality workflow documentation, we consider techniques which operate on different levels of documentation. Table 5.1 organises the techniques by the formalism they rely on to query workflows and by the level of detail at which workflows are queried. Text refers to techniques using natural language descriptions. RDF refers to techniques using light-weight semantic descriptions. OWL refers to techniques using rich semantic descriptions.

Using natural language descriptions. Three techniques were explored to exploit textual information in workflows. Firstly, a simple wrapper to the Google search engine API was built to investigate Google's performance on unprocessed workflow descriptions. Secondly, we adopted the existing Woogle service discovery tool for text-based service discovery (unrelated to Google) and adapted it into the Woogle4WF tool which translates a workflow to look like a service. Thirdly, to enable querying the internal structure of a workflow in conjunction with textual descriptions of its constituent services, we developed GUB4WF, a tool exploiting an existing graph matching library for sub-isomorphism detection.

Using light-weight semantic descriptions. We adopted the existing JMFeta tool for semantic service discovery. The tool relies on a similarity metric and annotations of services made based on the RDF(S) subset of the Resource Description Framework (RDF) language. It allows to query for similar workflows by providing it with workflow annotations that look like regular service annotations.

Using rich semantic descriptions. We analysed the possibilities and limitations of using the description logic based Web Ontology Language, OWL, for querying

Formalism	Technique not using workflow structure	Technique using workflow structure
Text	Google API, Woogle4WF	GUB4WF
RDF	JMFeta	
OWL		OWL4WF

Table 5.1: Summary of the considered automated discovery techniques.

the structure of data flows and proposed a novel *workflow ontology*. The combination of using selected example queries over the developed workflow ontology with the Racer description logic reasoner was dubbed OWL4WF.

5.1.3 Chapter structure

We organise the techniques of this chapter according to the three bottlenecks specific to workflow discovery identified in Chapter 3:

1. The discovery model
2. The process knowledge acquisition bottleneck
3. The ability to rank workflows

We regard the process knowledge acquisition bottleneck as the most pressing, *i.e.* the need to provide sufficient information to a discovery technique for it to work. Who will provide annotation to instantiate the discovery model in a global community? Who provides user training? Second, what kind of discoveries does the logical document view enable? No one technique is likely to cover all. Third, do the techniques allow for any rankings over workflows? The choice of discovery model determines what type of workflow information a technique requires and hence to a large extent determines the seriousness of the bottleneck.

Characterising the discovery model

Before characterising each individual technique, it helps to contrast the choice of discovery model made for each. To make such a comparison, we have resorted to a

	Index Terms	Full Text	Index Terms + Structure	Full Text + Structure
Retrieval	JMFeta	Google4WF**	OWL4WF*, GUB4WF**	Woogle4WF**

Table 5.2: Overview of workflow discovery tools in relation to document logical view.

classification of information retrieval tasks and logical views of documents from the information retrieval literature.

The classification in question is proposed in the standard work of [BYRN99] on information retrieval. The authors distinguish between the user task of *retrieval* and the user task of *browsing*. For reasons of scope, we focus on the retrieval task. Much can be said though in favour of a browsing approach to workflow discovery. This approach is complementary to the retrieval approach and lends itself to environments like a Web portal (e.g. `myExperiment.org`).

For the retrieval task, the techniques work either over a logical view of a document where a selection of words describes the text (*index terms*), a logical view where the collection of all words in the text represents the text (*full text*) or a view where a document is represented by its *full text and structure* (e.g. sections and chapters).

When it comes to workflow retrieval, we define the document logical view of *index terms* as any terms chosen by an (possibly automated) annotator to describe the workflow, *full text* as the entire collection of words that describe both the workflow and its services, and *full text and structure* as all the information available with full text, combined with any information about coordination of a workflow's services. Note that we do not assume a single representation language to represent this type of information. Candidate languages to capture workflow information include natural language, Semantic Web languages as well as traditional process formalisms such as Petri nets.

Consider the example workflow in Figure 4.18 on page 123, where a bioinformatics workflow is shown. Index terms in this example would be the bracketed terms that do not occur in the original workflow definition but instead were chosen by an annotator. *Full text* are all the words seen in the diagram, augmented with terms from different sources such as textual service descriptions by the service providers or ontology concepts written in OWL. *Full text and structure* in addition includes a detail of the control and data flow between services, including but not limited to the links shown in the diagram.

Table 5.2 relates the discovery model of each of the techniques to the document

logical views. In this chapter we cover each of the views. The table also indicates whether we have adopted a technique without change, designed it (indicated with "*") or designed and implemented it based on existing work (indicated with "**").

Before proceeding with a detailed discussion of each of the techniques, we relate them to workflow discovery matching types and existing work.

5.2 Related work

Table 5.3 shows the techniques presented in terms of the workflow match types identified in Chapter 3. We make the following observations about the relationship with the techniques covered in the related work section of Chapter 3 (page 91):

- None of the techniques presented here support complement-based discovery.
- In Sections 5.3 (page 138) and 5.4 (page 138) text-based approaches are presented to exploit the textual information captured in a workflow. Textual information is much easier to gather than formal annotation.
- Only two techniques provide detection of identical workflows down to the level of linked operations (the $SameW$ metric). One of these, GUB4WF, has been implemented as a prototype whereas the other, OWL4WF is only available as a proof of concept.
- Section 5.6 on page 141 investigates how workflow structure can be encoded into a workflow ontology based on the Web Ontology Language OWL. None of the techniques considered in Chapter 3 allow semantic discovery based on workflow structure.
- Finally, Section 5.7 on page 151 explores a graph matching optimization that alleviates the performance issues of graph-based approaches.

We are now ready to discuss each of the techniques in detail, based on their document logical view, process knowledge acquisition bottleneck and ability to rank workflows.

	GUB4WF	OWL4WF	Woogle4WF	Google4WF	JMFeta
Complement based					
ComplPar	no	no	yes	no	no
ComplOpExact	no	no	yes	no	no
ComplOpSuper	no	no	no	no	no
ComplOpSub	no	no	no	no	no
ComplOpOverlap	no	no	no	no	no
ComplOpEmpty	no	no	no	no	no
ComplWExact	no	no	no	no	no
ComplWSuper	no	no	no	no	no
ComplWSub	no	no	no	no	no
ComplWOverlap	no	no	no	no	no
ComplWEmpty	no	no	no	no	no
Identity based					
SamePar	no	yes	yes	no	yes
SameOp	yes	yes	yes	no	yes
SameParsOverall	no	yes	yes	no	yes
SameParsInternal	no	yes	no	no	no
SameOps	yes	yes	no	no	no
SameW	yes	yes	no	no	no

Table 5.3: A classification of the adopted workflow discovery techniques in terms of workflow match types.

5.3 Google4WF: Full Text

We choose Google [BP98] as an exemplar search engine doing full text search over XML documents. We constructed a wrapper, Google4WF, to use it for workflow discovery. Google4WF is implemented in Java and uses the Google API and the public Google index.

5.3.1 Knowledge acquisition bottleneck

A workflow often contains not only structural information about how to connect services, but also the name of the workflow, the workflow author, service names, input and output names, sometimes workflow descriptions and descriptions of the services used, etc. They are available by default in a workflow description. Since our workflow corpus is published as Scufl XML documents at myExperiment.org, it is available for indexing to the Google crawler.

5.3.2 Logical document view

We transformed each input workflow into a single query suitable as input to Google, by joining up all service and data names used in the workflow based on the logical OR construct. The use of the "related:" query prefix offered by Google, which returns URLs related to the input URL, did not work in practice.

5.3.3 Rankings

The results found by Google are from the myExperiment.org site directory where the corpus sample workflows are located. They are returned by querying the Google API. Since Google are a commercial enterprise, little detail is known about the ranking mechanisms in use other than a hyperlink-based popularity rating [BP98], which is of little relevance to us here. We include Google mainly for its status as a de facto benchmark for doing Web search.

5.4 Woogle4WF: Full Text + Structure

Workflows are not only software specifications. They are also documents which contain natural language. One can therefore imagine applying information retrieval on

workflow descriptions. To retrieve workflows, we have built a wrapper for the Woogle [DHM+04] tool called Woogle4WF. Woogle is a tool for similarity search for Web services based on standard information retrieval techniques such as the cosine measure, TF/IDF as well as a novel text clustering algorithm tailored specifically to the structure of Web service descriptions.

5.4.1 Knowledge acquisition bottleneck

Similar to Google4WF, the tool assumes the existence of a workflow as is, with no additional annotation. We wrote a parser to translate Scufl workflows in a form that the Woogle tool can interpret.

5.4.2 Logical document view

This technique regards a workflow as a bag of services. Different parts of the workflow (its natural language description, the constituent services with their descriptions and inputs, outputs) are kept as distinct categories during clustering. Structural information between services is discarded.

With Woogle4WF, essentially we establish a lossy translation of a workflow into the format of a Web service. We wrote a parser to translate Scufl workflows in a form that the Woogle tool can interpret. Whereas a Web service is characterized by constituent operations, a workflow has constituent services. Control flow and data flow between services is discarded. Woogle4WF maps a workflow into the Woogle WSDL service input format by regarding each workflow as a WSDL service and each constituent workflow service as a WSDL operation.

5.4.3 Ranking

The technique takes in a collection of Scufl workflows, clusters these in an off-line step, and then, when given an input workflow, produces rankings of workflows from the collection. The Woogle text clustering technique combines multiple sources of evidence to determine similarity [DHM+04]. In our case, it considers similarity between the textual descriptions of the workflow's services and of the entire workflow, and similarity between the parameter names of the consisting services. The key ingredient of the algorithm is a technique that clusters parameter names in the collection of workflows into semantically meaningful concepts.

5.5 JMFeta: Index Terms

We adopted, without change, a tool that does semantic similarity-based service retrieval. The tool, JMFeta, was developed by José M. Blázquez of Universidad Carlos III de Madrid in Spain during a research visit to the University of Manchester.

5.5.1 Knowledge acquisition bottleneck

Ontologies make use of logical expressions to denote concepts. They have been used with success in bioinformatics, library sciences, engineering and medicine. We regard ontological concepts as index terms. In the context of workflow discovery, an ontological representation of a workflow is needed. With respect to JMFeta, workflows are indexed in terms of service ontology concepts found in the ^{my}Grid ontology. We refer back to the discussion of service annotation based on the ^{my}Grid ontology in section 4.6.

5.5.2 Logical document view

We adopt the tool as an easy means to query for workflows that are described semantically as an atomic service, without any knowledge of its internal workings. JMFeta works over service annotations made based on the ^{my}Grid service ontology (discussed earlier in section 4.6).

5.5.3 Ranking

The tool computes the similarity between a query, expressed as a set of (attribute, value) pairs where the value is taken from the ^{my}Grid ontology.[1] The idea is that, given a query, the set of workflows is returned that match this query, sorted by similarity. The implemented (and as yet unpublished) semantic similarity metric relies on the notion of a semantic distance between the query and the results. Many of the results are obtained by query expansion and can be very specific in comparison to the original query. The matcher compares a query (which corresponds to a description where some of the attributes can be empty) with a set of workflows described as services. For every comparison, the matcher provides a value between 0.0 and 1.0 used to sort the results.

[1] This paragraph is due to personal communication with José M. Blázquez.

5.6 OWL4WF: Index Terms + Structure

Expanding on the view that a workflow can be seen as a semantically annotated service, in this section we develop a workflow ontology which extends the notion of a service ontology with structural connections between services.

5.6.1 The promise of OWL DL for service discovery

It is clear that to discover services, and workflows as a particularly complex type of service, one should describe and classify these in a manner appropriate for the searcher. Manually created classifications of services are inflexible and hard to manage when they become large, detailed and multi-axial. A description should always be self-coherent and consistent with respect to others in the classification. The service classification should evolve as the descriptions evolve, for instance when changes occur in the functionality or when additional known behaviour is added (i.e. one service can perform several tasks). To resolve this issue, one can keep service descriptions, classification and constraint management tightly coupled and treat this within the uniform framework of Description Logics (DL) [WSG+03], of which OWL DL [HPSvH03] is an exemplar, W3C standardised, language. Workflows, services and data types are then grouped into taxonomic hierarchies, together with definitions of the relationships and constraints between classes and their instances [MM03].

Various authors have experimented before with Web service discovery using DL reasoning, typically based on the OWL-S upper ontology ServiceProfile section, e.g. [SPAS03] or the Web Service Modeling Ontology (WSMO) DL Capability descriptions, e.g. [KLP+04]. We, however, are dealing with the discovery of workflows, and the difference between Web services and workflows puts additional requirements on the discovery task. We discussed how for discovery purposes a workflow can be seen as a service, and in this sense the existing work on service discovery applies. We also know from Chapter 3 that users want to look for workflows both as simple services and as composite services (service compositions). For simple service discovery, the existing work on discovering ServiceProfile or Capability descriptions can be used, which do not include control flow information. For composite service discovery queries over service flow and data flow, control flow information would be required. Even though detailed control flow information clearly is present in OWL-S and WSMO ontologies through the ServiceModel and Orchestration/Choreography descriptions, respectively, these parts of the ontologies are neither intended nor (to our knowledge) currently used

to support discovery. In this section we consider how OWL DL-based discovery works in the presence of simple control flow information. Before we start our discussion, we need a brief introduction to Description Logics.

5.6.2 Description Logics in a nutshell

Description Logics are expressive subsets of First-Order Logic, which lend themselves particularly well for describing domain knowledge. As explained in [BCM+03], pages 16-20, a Description Logic knowledge base is typically comprised by two components a T-Box and an A-Box. The T-Box contains general information about the problem domain (intensional knowledge, "classes"), whereas the A-Box contains knowledge that is specific to the individuals of the domain (assertional or extensional knowledge, "instances"). Intensional knowledge is usually thought not to change, whereas extensional knowledge is usually thought to be contingent, or dependent on a single set of circumstances, and therefore subject to occasional or even constant change. The basic task in constructing a T-Box terminology is classification, which amounts to placing a new concept expression in the proper place in a taxonomic hierarchy of concepts. Classification can be accomplished by verifying the subsumption relation between each defined concept in the hierarchy and the new concept expression. The basic reasoning task in an A-Box is instance checking, which verifies whether a given individual belongs to a specified concept. This is accomplished by verifying the subsumption relation between each defined concept in the (T-Box) hierarchy and the new concept expression. For further information, we refer to [BCM+03], Chapters 1 to 3, which provide an excellent introduction to Description Logics and associated reasoning.

The resulting classifications are lattices, not trees, as a description can have multiple parents. For example, in bioinformatics, a protein might be classified by what it transports, what it catalyses, the process it participates in; and where it is located. A service may be classified by its location, its cost, its inputs, its function and so on.

5.6.3 Knowledge acquisition bottleneck

When deploying an ontology-based solution, one needs to take the knowledge acquisition bottleneck seriously. Here we make two points specific to workflows.

Service classification is context specific

The same service used in different workflows will be linked to a different range of service classes depending on the task it performs in the workflow context. When annotating services in workflows and workflow fragments, it is not just a matter of how concrete services were once classified based on previous workflows; a service can be used for different tasks. Each use represents a different task description. One example is the multiple uses for the BLAST service. BLAST (Basic Local Alignment Search Tool) is a popular bioinformatics Web service for finding regions of genome sequence similarity (see `www.ebi.ac.uk/Tools/webservices`). A BLAST service can also be used as a plagiarism detection tool, by feeding it plain texts instead of DNA sequences. Hence one BLAST service potentially brings about multiple task descriptions.

Workflow semantics availability

In one sense, it is not hard to get a representation in OWL DL of a workflow: we can describe all its services as being of generic class Service and derive the appropriate links from the workflow specification. Obtaining a more useful representation is harder. Though progress has been made with tools like Feta [LAWG05], (even lightweight) semantic service annotations are proving hard to produce. In practice, bioinformaticians in ^{my}Grid resort to a combination of manual search, text search, and semantic search when discovering Web services, and build workflows that contain services with semantic descriptions and services without semantic descriptions. Workflows can come without any, or at best with incomplete, Web semantics, for the following reasons.

1. No matter how low the threshold is to providing annotation, if there is a threshold, some services and workflows will not get annotated.

2. Building workflows can take months, even years. One may want to keep other people, for example in a research group, informed of new, even incomplete, workflows and publish early results.

3. Workflows sometimes contain sensitive information related to Intellectual Property or refer to services that are unavailable outside an organization. In this case not all of the information in the workflow will be described for the outside world. The open world semantics of OWL DL allow to query for incomplete workflows.

5.6.4 Logical document view

We start by extending an existing T-Box service ontology [WSG+03] and then use this representation to answer a set of queries over a type of workflow fragments common in bioinformatics. We conclude by discussing the need for handling more complicated fragments and a hybrid approach to repurposing. Since our approach has not been implemented in a prototype, it cannot be evaluated based the human benchmark of the next chapter. To provide a simple proof of concept, we have instead selected seven queries designed to illustrate the strengths and weaknesses of the approach. We shall refer to the queries throughout the text.

Example queries

- Q1 Given a data point, service, fragment or workflow, where has this item been used before?
- Q2 Show the common data, services, service graphs or data graphs between two fragments or workflows.
- Q3 Given a set of data points, services, or fragments, have these been connected up in an existing base of workflows? If not, what are the closest available alternatives for doing so? How do these alternatives rank?
- Q4 As more and more workflows become available, fragments are reused and repurposed in a variety of workflows. How can one systematically keep track of these interrelationships?
- Q5 Which workflows are work in progress?
- Q6 Show the differences between two workflow versions.
- Q7 Show the evolution of a workflow over time.

Representing workflows / workflow fragments

Consider the Williams-Beuren syndrome gene annotation pipeline in Figure 5.1. Similar to many workflows we find in bioinformatics, this workflow is a pipeline that fans out: one starts out with a limited number of inputs, and ends up with many more outputs. We want to represent such a tree-like workflow in a DL. We assume that the T-Box collects generic descriptions of workflows which, given their generality, hardly change, while the A-Box provides a place for scientists and workflow developers to add new workflows.

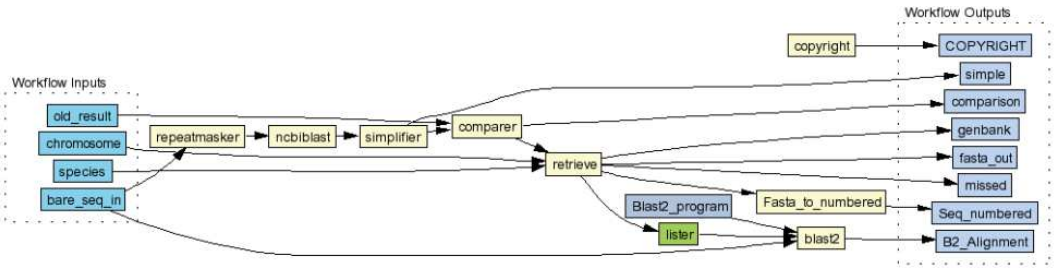


Figure 5.1: Williams-Beuren syndrome gene annotation pipeline.

T-Box Services are the basic building blocks of a workflow. We adopt the definition of a service used in the ^{my}Grid ontology [WSG$^+$03]. This ontology, originally in DAML+OIL and now in OWL Lite, contains 550 concepts and 69 roles (properties of concepts) and describes bioinformatics service classes. A `myGridService` service class (shown in what follows as `Service`) `usesOrProduces` one or more `BioDomainConcept`. Optionally, it includes a `performsTask` role relationship, as well as other, bioinformatics specific, roles (not shown).

In terms of notation, "$A \dot{\sqsubseteq} B$" indicates that B subsumes A. "$\doteq$" represents equality and "$\sqcap$" represents the Boolean AND operator. See [BCM$^+$03] for further details.

$$\begin{aligned} \texttt{hasInput} &\dot{\sqsubseteq} \texttt{usesOrProduces} \\ \texttt{hasOutput} &\dot{\sqsubseteq} \texttt{usesOrProduces} \\ \texttt{Service} &\doteq \texttt{BioProcess} \sqcap \exists \texttt{usesOrProduces.BioDomainConcept} \end{aligned}$$

We define workflows as entities that contain at least one service, by means of the transitive *has part* role `hp`.

$$\texttt{WF} \doteq \texttt{Process} \sqcap \exists \texttt{hp.Service}$$

Adding an ordering between services in a workflow description is made possible through the *has direct successor* role `hds`.

The following is a partial T-Box description for the sequence analyzer `SeqAna`

fragment combining three services of Figure 5.1.

$$\begin{array}{rcl@{\qquad}rcl} \texttt{SyntaxTranslator} & \dot{\sqsubseteq} & \texttt{Mediator} & \texttt{Mediator} & \dot{\sqsubseteq} & \texttt{Service} \\ \texttt{RepeatMasker} & \dot{\sqsubseteq} & \texttt{RemRedDNA} & \texttt{RemRedDNA} & \dot{\sqsubseteq} & \texttt{Service} \\ \texttt{BLAST} & \dot{\sqsubseteq} & \texttt{SeqSimSearch} & \texttt{SeqSimSearch} & \dot{\sqsubseteq} & \texttt{Service} \\ \texttt{BLASTn} & \dot{\sqsubseteq} & \texttt{BLAST} & & & \end{array}$$

Biologists often precede BLASTing a genetic sequence (*i.e.* running it through a BLAST service) with a RepeatMasker service to remove redundant structures and obtain a non-redundant sequence. We acknowledge the importance of this service combination for the biology domain by introducing a `BLASTNRSeq` workflow fragment in the T-Box. In general, concept expressions describing workflows are called *abstract workflows*.

$$\begin{array}{rcl} \texttt{BLASTNRSeq} & \doteq & \texttt{WF} \sqcap \exists \texttt{hp}.(\texttt{RepeatMasker} \sqcap \exists \texttt{hds}.\texttt{BLAST}) \\ \texttt{SeqAna} & \doteq & \texttt{WF} \sqcap \exists \texttt{hp}.(\texttt{BLASTNRSeq} \sqcap \exists \texttt{hds}.\texttt{Mediator}) \end{array}$$

The use of the `hds` and `hp` roles allows to derive that, for instance, `SeqAna` is subsumed by

$$\texttt{myFragment} \doteq \exists \texttt{hp}.(\texttt{RemRedDNA} \sqcap \exists \texttt{hds}.\texttt{BLAST})$$

We also wish to be able to deduce when one service (or piece of data) succeeds another even when a few services are in between (Q3). Introducing a role hierarchy has the desired effect. We define the *has successor* role `hs` to be transitive and a super role of `hds` by adding $\texttt{hds} \dot{\sqsubseteq} \texttt{hs}$ and $\texttt{Trans}(\texttt{hs})$ to the role hierarchy.

Moreover, it would be logical to expect that if a service, Service 1, is followed by a workflow fragment of which Service 2 is the first service, then Service 1 is succeeded by Service 2. To model such a derivation in a clean way, we would need to say that any *has successor* relationship between A and B followed by a *has part* relationship between B and C implies a *has successor* between A and C. This type of derivation can be achieved by means of a complex role inclusion axiom [HS03]. Unfortunately the role composition construct is unavailable in OWL 1.0 (in the OWL Lite and the OWL DL variant). See [DS04] for an overview of the constructs available in OWL.[2] We approximate the desired inference by making the *has part* role `hp` as a sub role of has successor `hs`. It remains to be seen in how far this approximation yields undesired

[2]It looks like the role composition construct will be available in the next version of OWL, version 1.1, in some part due to the use case provided here.

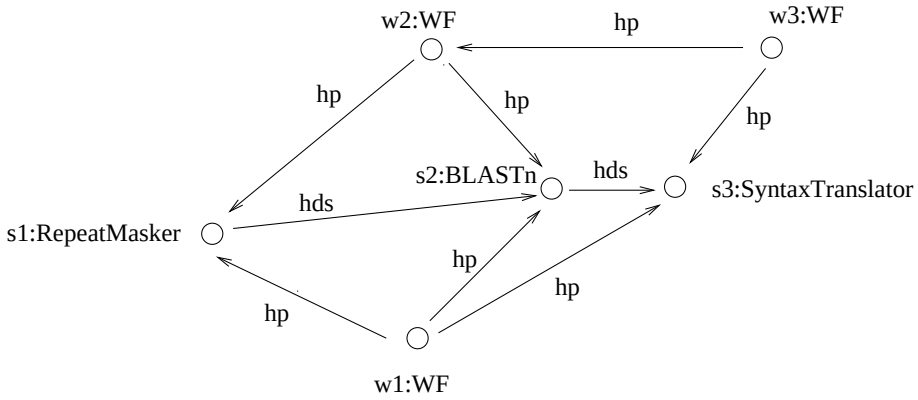


Figure 5.2: Contents of the A-Box without the hs roles, hp transitivity and inverses

effects.

Finally, the *has direct precursor, has precursor*, and *is part of* roles are modelled as the respective inverse roles of `hds`, `hs` and `hp`.

A-Box *Service instances* and *concrete workflows* in the A-Box instantiate the service classes and abstract workflows from the T-Box. As a guiding principle, abstract workflows are used for *structuring* the ontology, whereas concrete workflows are used for *query answering*. Concrete workflows are used for the annotation of snippets of working code, and this is what the user is interested in.

Below we give an example that shows part of an A-Box containing three concrete workflows and three service instances, based on the T-Box defined earlier. We omit the `hs` role assertions, `hp` transitivity and the inverse role assertions. Figure 5.2 represents the same information graphically.

Concept assertions

$$\begin{array}{ccc} \texttt{w1:WF} & \texttt{w2:WF} & \texttt{w3:WF} \\ \texttt{s1:RepeatMasker} & \texttt{s2:BLASTn} & \texttt{s3:SyntaxTranslator} \end{array}$$

Role assertions

$$\begin{array}{lll} \langle w1, s1\rangle : hp & \langle w1, s2\rangle : hp & \langle w1, s3\rangle : hp \\ \langle w2, s1\rangle : hp & \langle w2, s2\rangle : hp & \\ \langle w3, w2\rangle : hp & \langle w1, s3\rangle : hp & \\ \langle s1, s2\rangle : hds & \langle s2, s3\rangle : hds & \end{array}$$

Querying workflows and workflow fragments

With the T-Box and A-Box described so far, we now demonstrate some of the queries (relating to Q1, Q2 and Q3) that can be answered based on the representation developed in the previous section. We have used Racer[3] to support retrieval of fragments and follow the syntax of [CGL98]. We then consider the querying of incomplete workflows (Q4), as well as inexact fragment retrieval (Q3).

We rely on a Prolog-style syntax to write the conjunctive queries. "," represents Boolean AND, whereas ":-" represents an implication from the right-hand side of the formula to the left hand side.

Example queries for workflow fragments

Find the workflows that analyse a non-redundant sequence for similarity and then manipulate the results. Both queries are acyclic conjunctive. They return `w1` and `w3`.

$$\begin{aligned}(\mathtt{w}) \;&:-\; \mathtt{WF}(\mathtt{w}), \mathtt{hp}(\mathtt{w}, \mathtt{w}'), \mathtt{BLASTNRSeq}(\mathtt{w}'), \mathtt{hds}(\mathtt{w}', \mathtt{s}), \mathtt{Mediator}(\mathtt{s})\\ (\mathtt{w}) \;&:-\; \mathtt{hp}(\mathtt{w}, \mathtt{s}), \mathtt{hp}(\mathtt{w}, \mathtt{y}), \mathtt{hds}(\mathtt{s}, \mathtt{t}), \mathtt{hds}(\mathtt{t}, \mathtt{y}),\\ &\quad\; \mathtt{RemRedDNA}(\mathtt{s}), \mathtt{SeqSimSearch}(\mathtt{t}), \mathtt{Mediator}(\mathtt{y})\end{aligned}$$

Which workflows contain a service that does similarity search and a service that removes redundant information in DNA (the first query returns `w1`, `w2` and `w3`)? *Which workflows connect these 2 services* (the next query yields `w1`, `w2` and `w3`)?

$$\begin{aligned}(\mathtt{w}) \;&:-\; \mathtt{WF}(\mathtt{w}), \mathtt{hp}(\mathtt{w}, \mathtt{s}), \mathtt{hp}(\mathtt{w}, \mathtt{t}), \mathtt{SeqSimSearch}(\mathtt{s}), \mathtt{RemRedDNA}(\mathtt{t})\\ (\mathtt{w}) \;&:-\; \mathtt{hp}(\mathtt{w}, \mathtt{s}), \mathtt{hp}(\mathtt{w}, \mathtt{t}), \mathtt{hs}(\mathtt{s}, \mathtt{t}), \mathtt{RemRedDNA}(\mathtt{s}), \mathtt{SeqSimSearch}(\mathtt{t})\end{aligned}$$

Note the use of the `hs` role to indicate that intermediate links between `s` and `t` can be present. Neither Racer, the Manchester and Stanford OWL-QL implementations,[4] nor Pellet of U. Maryland[5] support the retrieval of the trace between role relations, *i.e.* to return the intermediate role relations linking two intermediate services (based on the `hs` and `hp` roles). Postprocessing the returned role assertions with a shortest path algorithm, in order to find the desired trace in a workflow, solves the issue.

[3] Web site: www.sts.tu-harburg.de/~r.f.moeller/racer/

[4] Web sites: www.cs.man.ac.uk/~glimmbx/ and onto.stanford.edu:8080/owql/FrontEnd

[5] Web site: www.mindswap.org/2003/pellet/index.shtml

Querying for incomplete workflows

The open world semantics of description logics allows to query for incomplete workflows (Q5). DLs allows to leave information implicit in a knowledge base – *e.g.* one can specify that at least one instance of a certain class exist, without having to name that instance.

Suppose that, in the example in Figure 5.1, a developer of a workflow has decided that a mediator service will be used, but has yet to decide where to put this service relative to the other services in the workflow. Even though the description is incomplete, it would still be of interest to other developers who are interested in the same type of mediation and thus it is useful to be able to publish and query such incomplete knowledge.

5.6.5 Ranking

So far, the queries involved the exact retrieval of fragments based on A-Box retrieval. One would also like to retrieve fragments that are largely relevant to a user (Q3) but happen to fall outside a strict subsumption relationship, *e.g.* the structure of two fragments is the same, except there are two services which are not in a subsumption relationship. A mechanism is needed to measure (dis-)similarity between fragments, calculating for instance how many services are to be moved, removed, added, replaced, merged or split to relate different fragments. It is key that such a mechanism is able to provide an explanation of why some fragments would be ranked higher than others.

DL role-based approaches and implementations relying on structural algorithms have been developed for $\mathcal{FL}^-$ in [BG98], which uses shared roles and role values for inexact matching, and in [BG99], which counts shared parent concepts. In [CCC+04] a structural algorithm based on abduction and contraction is presented for a fragment of $\mathcal{ALC}$. A tableaux algorithm for abduction and contraction based matching in $\mathcal{ALN}$ is presented in [CNS+04].

$\mathcal{FL}^-$, $\mathcal{ALN}$ and $\mathcal{ALC}$ are relatively inexpressive Description Logics compared to OWL. Directly applying the above approaches on the workflow ontology and the ^{my}Grid domain ontology has not proven possible, given the expressive constructs used in the workflow and domain ontology. In particular, for the workflow ontology, we need at least the combination of role hierarchies (indicated by the letter $\mathcal{H}$ in the name of the logic), role transitivity $\mathcal{S}$, inverse roles $\mathcal{I}$ and full existential quantification $\mathcal{E}$. If we want to be able to formulate questions that include ^{my}Grid domain concepts (such

as descriptions of input and output), nominals $\mathcal{O}$ and qualified number restrictions $\mathcal{Q}$ ought to be added.

In case no abduction algorithm for OWL is devised, approximation [BKT02] might offer a way out by simplifying the ontology in a non trivial way to the level of expressivity the abduction algorithm can handle. Another option is to stay within OWL and devise query relaxation strategies for a query manager.

Two alternative approaches to handle similarity in DL combine elements from other paradigms, thereby creating a hybrid formalism. The *vector-based* approach adopts normalised vectors and the cosine measure from information retrieval, *e.g.* [MSST93], whereas the *probability-based* approach tries to merge Bayesian inference with DL reasoning, *e.g.* [KLP]. The theoretical foundation and practical implications of these alternative approaches are less well understood. It is unclear how they would provide sufficient explanation facilities as to why a particular ranking was constructed.

5.6.6 Outlook

Capturing data flow and more complex control flow The current representation largely ignores data flow. One could introduce data objects that have input and output relationships in the A-Box. For control flow, due to the lack of expressive power of OWL, we must abstract from the structure of the concrete workflows when they are described in the T-Box. So far, we only use `hds` to represent control flow. For the fragment discovery purposes of scientists, this is probably what one wants to do anyway: these users are not interested in intricate control flow details. There may be cases where more complex control mechanisms such as loop, conditionals or concurrency constraints need to be modelled. We can easily capture part of a conditional in the T-Box by introducing *e.g. has possible successor* roles. With respect to loops, one cannot define and query for *loops* in the abstract workflows. One is still able to query for loops over concrete workflows in the A-Box.

A hybrid approach to repurposing It is clear that, when building workflows, we are often confronted with various problems that can (and should) be solved "syntactically" such as versioning support (Q6 and Q7). Moreover, enacting workflows generates data points which people query for. It seems sensible to represent such objects not in the A-Box but in a database, as in effect one has generated everything there is to know about a particular enacted workflow and no extra inferences can be drawn.

5.7 GUB4WF: Index Terms + Structure

If one regards the service names of a workflow as index terms and keeps the structure intact, a different technique becomes possible, which is not dependent on annotations in formal Knowledge Representation languages.

5.7.1 Knowledge acquisition bottleneck

A graph matching technique can be applied on workflows without a need for additional annotation if it assumes only the existence of the workflow specification.

We chose a graph as the data structure and an algorithm for graph sub-isomorphism detection as the basis of the matching algorithm. Our choice of a graph as the data structure for capturing a workflow is motivated as follows. Firstly, multiple scientific workflow environments have adopted a graphical representation to present workflows to end users. For these environments, the choice of a graph as the vehicle for matchmaking potentially makes explaining the discovery process and its results more intuitive. Taverna workflows, too, are visualized as graphs. Taverna's graphs are directed and acyclic - this follows from the semantics of Taverna's Scufl language, which correspond to a lambda calculus with a list monad [Tur06]. Secondly, a graph is a very expressive data structure. The benefits and drawbacks of different exact and inexact graph retrieval algorithms and heuristics are well understood. Finally, graphs provide a theoretical underpinning for the Resource Description Framework (RDF), a W3C recommendation for describing semantic information on the Web. Techniques we adopt for workflow diagrams based on graphs potentially extend to RDF graphs describing workflows. Currently, in ^{my}Grid over 500 biological Web services are annotated in RDF(S)[6] and the transition path to workflows appears natural.

5.7.2 Logical document view

Graph matchers assume graphs of a certain kind as input; in our case, the graph matcher works over attribute-less graphs of nodes and directed, attribute-less edges. The contents of a graph impacts the outcome of the matching process. Our generated graphs include the workflow's overall input and output as nodes as well as intermediate graph nodes instantiated with the names of the services connecting the workflow's input and output. The graph's edges are defined as the connections between the services.

[6]Web site: `http://www.mygrid.org.uk/feta/mygrid/descriptions`

5.7.3 Rankings

To produce results, the graph matcher relies on sub-isomorphism detection ("subgraph matching"). The technique for graph sub-isomorphism detection we restorted to is optimized to work over a repository of graphs. The technique was developed and implemented by Messmer and Bunke [MB00]. Standard methods for sub-isomorphism detection usually work on only two graphs at a time. However, when comparing workflows, there is more than one graph in the repository that must be matched with the input graph. Consequently, it is necessary to apply the subgraph isomorphism algorithm to each pair of repository graph - input graph, resulting in a computation time that is linearly dependent on the size of the repository. Messmer and Bunke's approach is based on a compact representation of the repository graphs that is computed off-line. The representation is created by decomposing the repository graphs into a set of subgraphs, where common subgraphs of different graphs are represented only once. During on-line matching, they are matched exactly once with the input graph, yielding a technique that is only sub-linearly dependent on the number of the graphs in the repository [MB00]. As explained below, the optimization is currently unavailable in our tool, but remains of interest.

The data structure and graph matching algorithm in itself are not sufficient to create workflow rankings. The translation of a workflow into a graph and the way results are ranked and presented have a great impact on results. These steps need further explanation.

Graph Parser

The Parser translates the Scufl specification of all workflows in a form and format suitable for the graph matcher. Our chosen graph matcher is rather inexpressive and only accepts attribute-less graphs of nodes and (directed or undirected) attribute-less edges. The contents of a graph impact the outcome of the graph matching process, typically as a trade-off between accuracy and performance.

The following steps occur in our translation from a Scufl workflow to a graph:

1. The workflow's overall inputs and outputs are included as named nodes in the graph.

2. The intermediate nodes are instantiated with the names of the services connecting the workflow's input and output, while ignoring all information about intermediary inputs and outputs.

3. The graph's edges are defined as the connections between the services.

Therefore, the information that is captured roughly mirrors the type of information available in the workflow diagram in Fig. 5.1. The colouring which indicates service type has been lost in translation, though. We also lost the distinction between overall inputs, overall outputs and services and conflated them into nodes.

Other parsing strategies are possible, for example based on including intermediary input and output names. The inexpressivity of the graph matcher means we cannot straightforwardly combine intermediary inputs and outputs with information about the service's name. Judging from our corpus, relying solely on intermediary input and output names would seem careless, given that they often consist of generic terms such as "value" or "query."

Graph Matcher

With the workflows turned into graphs, Messmer and Bunke's graph matcher can be put to use. The implementation supports efficient matching of a large input graph to a collection of smaller graphs in the repository. In the context of workflow retrieval, this corresponds to the case where users would want to retrieve those workflows in the repository which correspond to a fragment in the user's workflow, perhaps to find out which other authors did the same analysis.

The case where one starts out with a small input graph and matches this to a collection of larger graphs is not implemented by the prototype. Unfortunately, in the case of workflow matching, the latter case, where one starts out with a small exemplar workflow and one would like to compare it against a repository of large, finished workflows, seems of more practical relevance. Re-implementing the graph matching algorithm to cater for this scenario is non-trivial, however, and beyond our scope. Instead, we resorted to inverting the matching process by sequentially treating each of the (large) repository graphs as an input graph to the matcher, and treating the (small) exemplar workflow as the whole repository. This work around destroys the graph repository optimization since it treats the prototype as a standard subgraph matching package which is invoked as many times as there are workflows in the repository.

HTML Formatter

The results from the Graph Matcher are rendered into an HTML page, which specifies how many nodes are shared between workflows and what their size difference is.

It also contains links to the workflow specifications. Furthermore, we highlight the differences between the input workflow (`AffyidToBlastxPDB.xml`) and the retrieved ones. Fig. 5.3 shows the example where the joint use of the Blastx service is detected by the matcher. The example also illustrates how the graph matcher, based on our translation, conflates overall inputs and outputs and services. As it cannot not distinguish between "Blastx" as a service and "Blastx" as a workflow output, the "Blastx" output is wrongly highlighted in the repository workflow, even though no output with exactly the string "Blastx" is present in the input workflow.

5.8 Summary

In this chapter we gave an overview of workflow design discovery techniques modelled to work over data flow workflows that are built with the Taverna workflow editor. The presented techniques are novel either because (i) the adaptation of an existing search engine to the workflow discovery problem is new or (ii) because they were developed from scratch specifically for structural workflow discovery (based on well-known algorithms from the graph matching and description logic reasoning literature).

In Chapter 6, we will apply these techniques on the benchmarks generated in Chapter 4, in order to evaluate their worth on practical discovery tasks. A discussion of possibilities for future work is covered in the future work chapter.

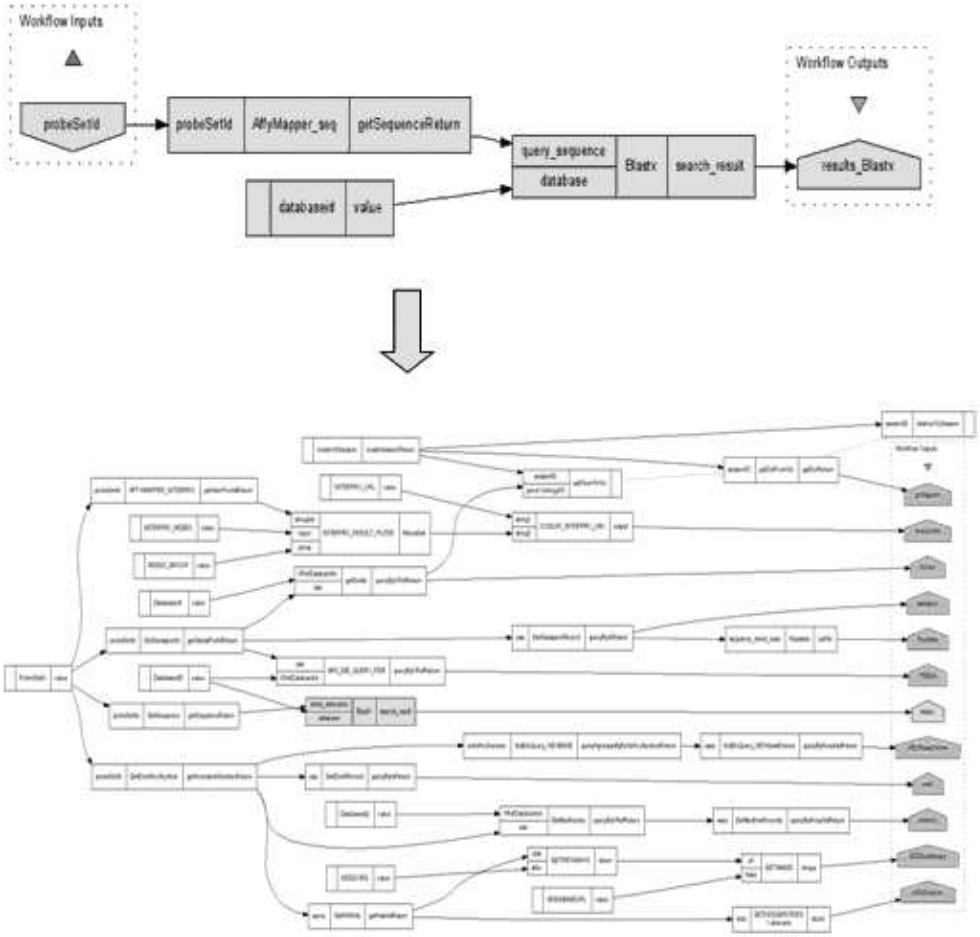


Figure 5.3: Visualization of overlap between the input workflow and one in the repository.

Chapter 6

Evaluation of discovery techniques on benchmarks

We evaluate the performance of the techniques by comparing their solutions on discovery tasks with the solutions provided by bioinformaticians when participating in the user experiments of Chapter 4. Several choices were made as part of the evaluation method. After making these explicit, we present the results of the evaluation.

6.1 Evaluation method

We made several pragmatic choices in performing the evaluation. Ideally, one would measure the performance of all techniques by comparing them against the solutions of participants on all performed workflow discovery tasks for those user experiments which yielded statistically coherent results. In practice, we evaluated only particular combinations of techniques and data sets. Table 6.1 lists these combinations. The techniques are either (i) fully evaluated against the assessments of all participants on all tasks in a data set, (ii) evaluated partially against the assessments of all participants for the case of an individual task, or (iii) not evaluated on a particular data set. The lack of full evaluation is due to three reasons:

1. *Unavailability of annotation.* If a set of workflows as used in a user experiment is not annotated, a semantics-based technique cannot be tested on that set. The 18 workflows of benchmark 3 are semantically annotated only to a degree: only its constituent services and not structural connections between its services. This allowed to still use the JMFeta tool on this set. Given the lack of suitable annotation, OWL4WF was evaluated in the previous chapter based on a set of sample

Data set	Discovery task	Google API	Woogle4WF	GUB4WF	JMFeta	OWL4WF
Experiment 1	Similarity - cross author			Case		
Benchmark 1	Similarity - personal		Full	Full		
Benchmark 2	Similarity - cross author	Case	Full	Full		
Benchmark 3	Similarity and complement - cross author				Case	
None	Similarity - personal and cross author					Case

Table 6.1: Summary of evaluation method for automated discovery techniques

queries.

2. *Prediction of performance.* During the evaluation of techniques, we would encounter fundamental reasons why a particular technique would perform badly beyond a particular discovery task. For reasons of time constraints, in these cases, we decided to stop exploring its performance further instead of confirming whether this predicted poor performance actually holds across all available data sets.

3. *Processing of benchmark results.* With respect to benchmark 3, we were unable to process results in time in order to use them in conjunction with the Woogle4WF tool.

In addition, in one case we rely on a statistically unconvincing data set. The potential and limitations for producing workflow rankings based on the GUB4WF tool are partially demonstrated based on data from user experiment 1.

With these considerations in mind, we partially evaluate the following discovery tasks:

1. Similarity-based personal and cross-author discovery.

2. Complementarity-based discovery.

6.2 Similarity-based personal and cross-author discovery

We use the data from experiment 1 and benchmarks 1-3 to explore the performance of the tools on similarity-based personal and cross-author discovery.

6.2.1 Results based on data from Experiment 1

Given the lack of a robust benchmark based on experiment 1 (see page 108), no generic claims can be made as to how useful any tool is for end users. As a simple showcase, instead we aim to have the GUB4WF tool replicate the averaged workflow similarity ranking of Table 4.3 on page 109 with respect to the exemplar used in the user experiment.

The averaged similarity ranking, in decreasing order, with respect to the exemplar is: (i) Workflow 1; (ii) Workflow 4; (iii) Workflow 3; (iv) Workflow 2; and (v) Workflow 5 (see section 4.2.5 for the workflow diagrams).

Running the GUB4WF Parser on the 89 workflows takes about ten seconds on a PentiumIV/512MB RAM/WindowsXP machine. The matching process itself takes about five seconds. The graph size ranged from three nodes to 36 nodes.

Different ranking strategies can be adopted. In terms of ranking the graphs retrieved by the graph matcher, we have experimented with three factors: the similarity in terms of overall input, overall output and internal services shared between workflows (i.e. nodes shared between graphs), the difference in workflow size (number of nodes) and a simple lexical transformation (uncapitalization on the nodes). We show the impact of three different strategies for the above showcase:

1. *Shared nodes, string matching.* The Graph Matcher always returns the biggest subgraph found during matching with the input. We use the size of this subgraph as a measure to rank the collection of matched workflows. Without manipulating the names of the nodes (workflow input, output and services), this matching strategy returns 9 results, and of the list to be ranked contains workflows 1 and 2 (listings are provided on-line).

2. *Shared nodes, lowercase string matching.* When adapting the above strategy to make all node name assignments lower case during the Parser process, another 14 workflows show up in the matching results, including workflow 4. The list now includes workflows 1, 2 and 4, but wrongly ordered.

Results from the workflow discovery matcher using the following repository:

Local directories:
C:\MyWorkflows

Web directories:
http://workflows.mygrid.org.uk

Based on your current workflow, the following **workflows** might be of interest:

AffyidToBlastxPDB.xml: 0 node(s) bigger, 5 node(s) shared
BlastXAgainstAffyIDSequence.xml: 0 node(s) bigger, 5 node(s) shared
AffyidToGeneAnnotation2.xml: 9 node(s) bigger, 2 node(s) shared **workflow 1**
AffyidToFastaSequence.xml: -2 node(s) bigger, 1 node(s) shared **workflow 4**
AffyidToEmblRecord.xml: -1 node(s) bigger, 1 node(s) shared
AffyidToMedlineIds.xml: 0 node(s) bigger, 1 node(s) shared
AffyidToBlastnHtgoMus.xml: 1 node(s) bigger, 1 node(s) shared
AffyidToMedlineRecords.xml: 1 node(s) bigger, 1 node(s) shared
workflow.xml: 2 node(s) bigger, 1 node(s) shared
AffyidToProteinAnnotationPipeline.xml: 3 node(s) bigger, 1 node(s) shared
AffyidToGoGraph.xml: 5 node(s) bigger, 1 node(s) shared
AffyidToGeneAnnotationPipeline1.xml: 6 node(s) bigger, 1 node(s) shared
AffyidToGeneAnnotationnoGOIDandPepstats.xml: 11 node(s) bigger, 1 node(s) shared
AffyidToGeneAnnotation4.xml: 12 node(s) bigger, 1 node(s) shared **workflow 2**
AffyidToGeneAnnotation10.xml: 15 node(s) bigger, 1 node(s) shared
AnnotationPipelineNoKeggForSEEK.xml: 18 node(s) bigger, 1 node(s) shared
AffyidToGeneAnnotation3.xml: 18 node(s) bigger, 1 node(s) shared
AffyidToGeneAnnotationforPaper.xml: 20 node(s) bigger, 1 node(s) shared
AnnotationPipelineForSEEK.xml: 26 node(s) bigger, 1 node(s) shared
NFKBIE_AFFYID_TO_ANNOTATION_alternate.xml: 32 node(s) bigger, 1 node(s) shared
NFKBIE_AFFYID_TO_ANNOTATION.xml: 32 node(s) bigger, 1 node(s) shared
AffyIdToGeneAnnoation(NFKBIE)_final.xml: 40 node(s) bigger, 1 node(s) shared
ProbeIDBasedDataMining.xml: 46 node(s) bigger, 1 node(s) shared

Figure 6.1: Output for ranking strategy 3 with respect to the exemplar workflow.

3. *Shared nodes, lowercase string matching, size.* Introducing a measure that compares the size of the exemplar workflow to the comparison workflow ranks workflows 1, 2 and 4 in the right way. We show the results of this strategy in Fig. 6.1. Workflows are ordered in the first instance by the number of nodes they share with the exemplar, and, in those cases where two workflows in the list have the same number, they are ordered by the size of the difference between the two workflows.

Strategy 3 still fails to retrieve workflows 3 and 5. Upon closer inspection, it becomes clear that inexact string matching of the service names, or information retrieval techniques in general, could offer a solution for these cases. Another solution would be matching based on classes of similar services, which opens up the door for semantic annotation.

6.2.2 Results based on Benchmarks 1 and 2

The Google API, Woogle4WF and GUB4WF are selectively used on benchmarks 1 and 2 (Table 6.1).

Google4WF

The combination of the Google API based wrapper and the proprietary ranking mechanism used by Google on the raw XML files in which the Taverna workflows are captured yields poor results for Benchmark 2. This makes clear that the approach would not work for the other benchmarks either. A rendering of a workflow into a Web page friendly format, including linkage of workflow Web pages to Web pages representing constituent services, is likely to improve rankings.

GUB4WF and Woogle4WF

Testing the GUB4WF graph matcher and the Woogle4WF text clustering technique on benchmarks 1 and 2 (see experiments 3 (page 114) and 4 (page 116), respectively) yields more interesting results.

In addition to the raw performance of these two techniques, we also consider the "combination hypothesis" as an additional technique – the idea, coined by IBM, that further advances in search technology will be based on a cross-disciplinary approach. In our context, we consider the impact of combining the results of the graph matching

Task	Measure	Top x results	Graph matcher	Text clustering	Intersection	Union
Versioning	Precision	25	65	34	51	44
	Recall	25	50	24	17	57
	Precision	10	65	35	90	48
	Recall	10	21	9	7	25
	Precision	5	70	40	83	56
	Recall	5	12	6	2	16
Cross author discovery	Precision	11	-	60	-	-
	Recall	11	-	74	-	-
	Precision	5	-	50	-	-
	Recall	5	-	36	-	-

Table 6.2: Average recall and precision for versioning and cross-author discovery on Benchmarks 1 and 2.

and text clustering techniques. We identify two options: (i) use the *intersection* of results (when both techniques agree) or (ii) use the `union` of results.

Table 6.2 summarises the performance of the 2+2 techniques. The first band of figures shows the average precision and recall for performing the 2 personal discovery, *i.e.* versioning tasks. The second band of figures gives the average precision and recall for cross-author discovery for the 11 input workflows (12 minus the one dummy). It shows performance with respect to the top x results returned by a given technique (values in percentage; higher is better).

The figures bring out the trade-off between precision and recall, in that an increase in precision means a decrease in recall. The only exception to this is the performance of the text clustering, which might be explained by the relative small set of 21 workflows over which the clustering algorithm operated.

The different classes of discovery techniques come with their own strengths and weaknesses. The *text clustering* technique performs well on cross-author discovery, but does poorly when it comes to versioning. The *graph matcher* does well in comparison on the versioning task, but when applying the graph matcher for cross-author discovery, no results are returned in any of the cases. As a result, the application of the combination hypothesis turns out to be sensible only in the case of versioning, where both techniques yield results. The *intersection* technique has good precision on the versioning task compared to the other techniques, but displays a drop in recall, whereas

the *union* technique displays a converse pattern. We conclude that the combination hypothesis idea does not improve the quality of search results overall in our experiment; one has to choose between either bettering precision or bettering recall.

We observe that the figures leave much room for improvement. This suggests that changes in the approach are needed, be it in the form of a more refined graph matcher, better natural language descriptions of the workflows or indeed through semantic annotation. Comparing the results of the techniques and the experts, we found multiple matches which were only identified by the experts. These missed matches relied mainly on expert background knowledge of the biology and bioinformatics behind the services. One such example is illustrated in Figure 6.2. These workflows represent different outcomes but share the same semantic output type (shown as a boxed ontology concept from the ^{my}Grid service ontology). WF1 produces an output of SwissProt ids from the getSwissprotIds Web service, while WF2 outputs these at the end of the workflow. This implicit link is important because the goal of the author of WF2 could be to extend her workflow to accomplish the same as WF1, namely to retrieve Gene Ontology IDs.

6.3 Complementarity-based discovery

Benchmark 3 (page 117) is the only one to capture the performance of human experts on tasks that require assessment of workflow complementarity.

None of the tools we discussed in Chapter 5 are built to support complementarity. In collaboration with José M. Blázquez of Universidad Carlos 3 de Madrid in Spain, we tested what the JMFeta tool, not built for this purpose, would achieve. We established that the JMFeta tool does not predict the correct answer on the simple example exercise designed for the participants of user experiment 5 (see Appendix C). To solve the task, only overall inputs and outputs of all workflow were used. This confirms that knowledge of the internals of the workflow would be needed. The Woogle4WF tool in that respect may do slightly better on the same task given that it also indexes the constituent services of a workflow; future experimentation would need to confirm this.

6.4 Summary and discussion

All the data set based evaluations provide evidence that the discovery techniques under consideration are highly task specific. This suggests that, *to support the full range of*

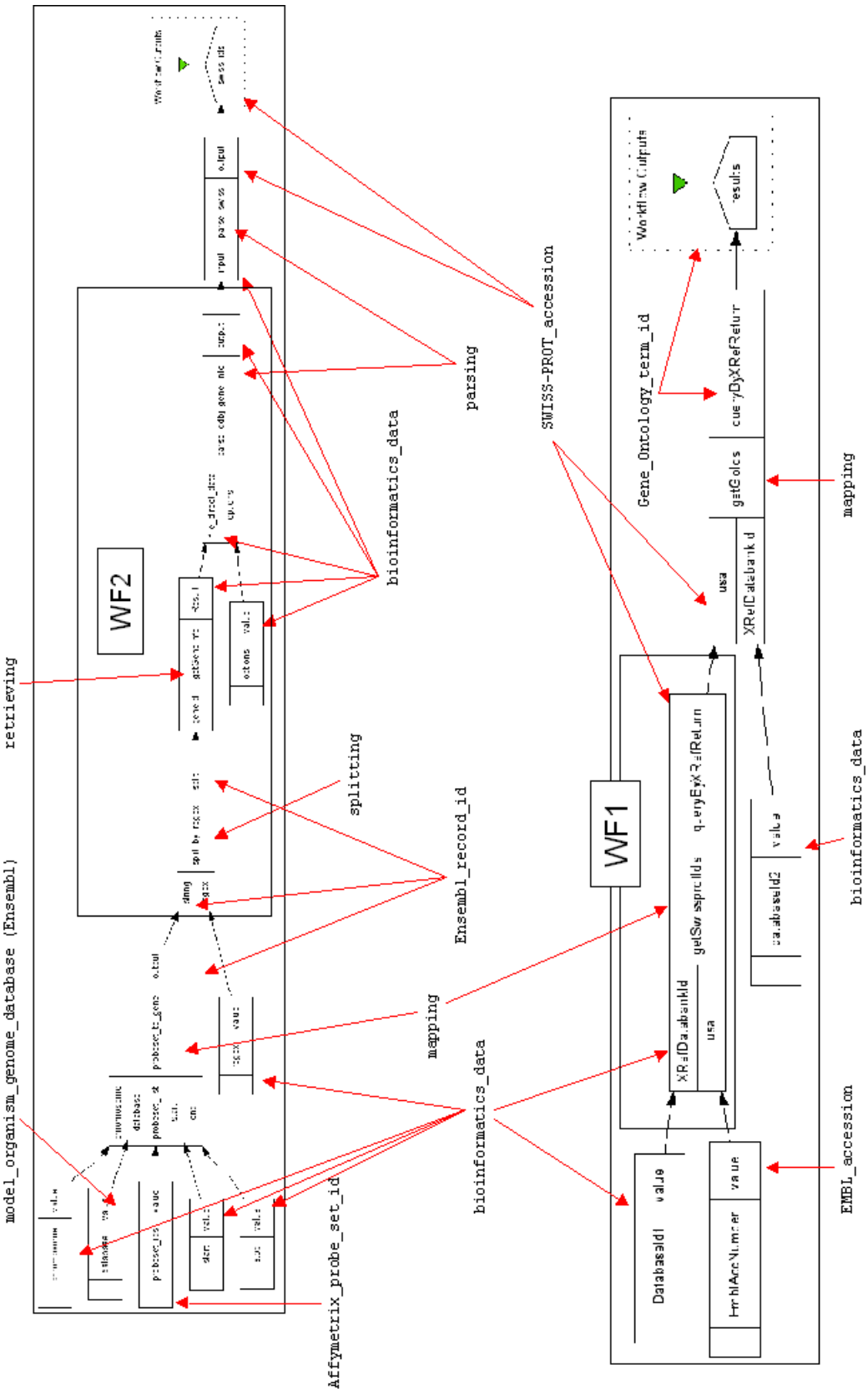


Figure 6.2: Background knowledge in workflows.

discovery tasks, a suite of techniques will be needed. The results for the techniques on the different discovery tasks are summarised as follows:

Similarity-based personal discovery. The use of a graph matcher in GUB4WF to represent workflow structure proved valuable to support personal discovery (GUB4WF + Benchmark1; GUB4WF + Experiment1). Conversely, the lack of a structural matching component in the Woogle4WF tool meant it was unable to detect any structural relations between workflows which resulted in poor precision (Woogle4WF + Benchmark1).

Similarity-based cross-author discovery. The use of a text clustering component in Woogle4WF proved valuable to support cross-author discovery (Woogle4WF + Benchmark2; Woogle4WF + Benchmark3). Conversely, the lack of a text clustering component in GUB4WF meant it was unable to detect results during cross-author discovery (GUB4WF + Benchmark2; GUB4WF + Experiment1). Finally, the combination of the Google API based wrapper and the proprietary ranking mechanism used by Google for unprocessed workflows yielded poor results for cross-author discovery (Google API + Benchmark2).

Complementarity-based discovery. The performance of applying light-weight semantic and text-based similarity-based techniques on complementarity-based tasks was poor (Woogle4WF + Benchmark3; JMFeta + Benchmark3). The results from the limited case study based on the JMFeta tool (JMFeta + Benchmark3) suggest that *the indiscriminate use of lightweight semantics offers little benefit over the indiscriminate use of the Woogle4WF text-based technique*.

In the next chapter of the thesis, we will revisit the fundamental assumption made about scientific workflows in the beginning of the thesis. We assumed that such workflows are implemented using only one kind of model of computation. The chapter will lift this restriction and will investigate the impact this has on workflow re-use.

Chapter 7

Re-use of Models of Computation

Scientific workflow environments typically offer support for the design, enactment and provenance recording of computational experiments. In Chapter 2's discussion of workflow systems, we observed that support for re-use and discovery is mostly lacking.

Thus far, in our discussion of re-use and discovery, we have assumed that the model of computation (MoC), or the formal abstraction of computational execution, is fixed. We envisage a future where computational experiments are modeled based on different models of computation, however.

When the assumption of a single model of computation is removed, the question of workflow re-use becomes more complex. Can models of computation be combined in all possible combinations? Are some more re-usable than others? This chapter investigates how workflow re-use is affected when multiple MoCs are available.

7.1 Problem statement

Chapter 2 described several application scenarios in e-science for composing workflows based on diverse models of computation (see page 29). To date, little is known about how models of computation are joined.

Despite the conceptual appeal of the listed scenarios, few compositions of models of computation have made it into e-science practice. There are two main reasons for this. First, workflow tools have not generally supported heterogeneous MoCs, so to achieve it, workflow designers would need to combine distinct frameworks. The ensuing software configuration complexity and fragility, the diversity of modeling syntaxes, and the conceptual complexity of the resulting models are all daunting. "Tool integration" (the combination of distinct modeling and workflow systems) is usually not the

best way to accomplish our objectives. Second, using heterogeneous MoCs requires e-scientists to think of their applications in a different way, and little exists in the way of training materials, guidelines, or examples. The Ptolemy project has had as its objective all along the first of these two problems [BHLM94] [EJL+03]. This chapter addresses the second concern.

7.2 Related work

Regarding the second concern, determining which MoC combination is the most appropriate for a particular scenario is potentially a difficult exercise. Designers need to understand nuanced distinctions between the MoCs. This task is complicated by the lack of a standardized representation to specify the assumptions and requirements of MoCs for composition. Prior work has offered formalisms for describing MoCs [MB07] [HB07] [BBS06], and comparing them [LSV98] but less for their composition. In addition, workflow systems typically do not provide a formal characterization of the used MoC(-s). Often, MoC behavior is buried inside legacy software components (e.g. ODE solvers in optimized Fortran libraries). This makes it harder to model and execute interactions between MoCs. Finally, a systematic study of MoC compositions is lacking. Knowledge about how models of computation are joined is scattered across communities.

Regarding this latter point, within the scientific workflow system community, transformations of scientific data are commonly modeled in dataflows [GRD+07]. Several systems are experimenting with mixing dataflows with control flows. The authors of [BLNC06] use Kepler [LAB+05] to model control-flow intensive subtasks inside of data flows. Similar to the approach of [BLNC06], Taverna [OGA+05] treats policies for individual services (e.g. retry strategies) differently to the overall (lamda calculus based) data flows [Tur06]. At a higher level, both Kepler and Taverna enable computational steering of data flows, as does Inforsense [CGWG07]. Within the business workflow systems community, work exists on how Petri nets are to be combined with other formalisms [vdA05]. MoC compositions are also investigated in the hybrid systems community. Hybrid systems are dynamic systems that exhibit both continuous and discrete dynamic behavior. Compositions of models of computation in this case involve the modeling of time. Within the software engineering community, UML Statecharts, rooted in the work by Harel [Har87], combine two MoCs, finite state machines (FSMs) and synchronous/reactive (SR) models [BB91]. This has been generalized to

show that FSMs can be usefully combined with a number of concurrent MoCs, resulting in the notion of modal models [GLL99]. In the embedded systems community, a number of researchers have explored mixed MoCs for software [JS05] and hardware [PS04] system designs.

The approach we assume in this chapter constructs workflows as actor-oriented models. Our notion of actor-oriented modeling is related to the work of Agha [Agh86] and others. The term actor was introduced by Hewitt to describe the concept of autonomous reasoning agents [Hew77]. The term evolved through the work of Agha and others to describe a formalized model of concurrency [Agh86]. Agha's actors each have an independent thread of control and communicate via asynchronous message passing. We have further developed the term to embrace a larger family of models of concurrency that are often more constrained than general message passing. Our actors are still conceptually concurrent, but unlike Agha's actors, they need not have their own thread of control. Moreover, although communication is still through some form of message passing, it need not be strictly asynchronous.

Actor-oriented modeling has been around for some time, and is in widespread use through such programs as Simulink, from The Mathworks, LabVIEW, from National Instruments, and many others. It is gaining broad legitimacy in software engineering through the efforts of OMG in UML-2, for example. Many research projects are based on some form of actor-oriented modeling, but the Kepler/Ptolemy II approach is unique in the breadth of exploration of semantic alternatives and in its ability to combine distinctly different actor-oriented MoCs.

7.3 Workflows and Hierarchy

We situate our discussion in the context of the Kepler [LAB+05] and Ptolemy II [EJL+03] environments. In contrast to Taverna [OGA+05], Kepler is a scientific workflow environment that allows scientists to choose from multiple MoCs. Kepler is based on the Ptolemy II simulation environment. Kepler and Ptolemy II also allow for workflows that contain multiple MoCs. This chapter explains how MoCs are combined in Kepler and Ptolemy II and analyses which combinations of MoCs are currently possible and useful. It demonstrates the approach by combining MoCs involving dataflow and finite state machines. The resulting classification should be relevant to other workflow environments wishing to combine multiple MoCs.

Kepler/Ptolemy II comes with a wide range of MoCs, which are implemented as so

called *directors*. We first introduce the notion of hierarchy as the key concept for mixing MoCs in a workflows. Next we provide an overview of MoCs in Kepler/Ptolemy II. For a scientific workflow developer, determining which MoC combinations are legal is non trivial. This leads us to establish MoC compatibility, based on the notion of actor abstract semantics and presents a classification of MoCs combinations. Finally, we discuss the validity of the approach and demonstrate successful and unsuccessful combinations of dataflow and finite state machines.

Ptolemy II is a Java-based environment for heterogeneous modeling, simulation, and design of concurrent systems. Ptolemy II forms the core of Kepler [LAB+05], an environment for building scientific workflows. The focus of Ptolemy II is to build models based on the composition of processing components called *actors* [Agh86]. Actors are encapsulations of parameterised actions performed on input tokens to produce output tokens. Inputs and outputs are communicated through ports within the actors. They provide the common abstraction used to wrap different types of software components, including sub-workflows, Web and Grid services.

The interaction between the actors is defined by a Model of Computation. The MoC specifies the communication semantics among ports and the flow of control and data among actors. *Directors* are responsible for implementing particular MoCs, and thus define the "orchestration semantics" for workflows. By selecting the director, one selects the scheduling and execution semantics of a workflow. Many actors can work with several directors, adapting their behaviors to match the semantics of the director [LAB+05]. The models of computation implemented in Ptolemy as directors are described in detail in Vol. 3 of [BLL+05]. A subset of them, including dataflow, time and event dependent directors, is available in Kepler. Key to mixing MoCs in a workflow is the notion of hierarchical abstraction. Figure 7.1 shows a Kepler chemistry workflow using the PN director, which implements a process networks MoC [SAB06]. This workflow contains a *composite actor* (a.k.a. sub-workflow) named Babel. The implementation of Babel actor is another workflow that contains another director, the SDF director, which implements a synchronous dataflow MoC. This example mixes two MoCs in a single, hierarchical workflow.

In Ptolemy II/Kepler, hierarchy can serve either of two roles. First, it can be simply an organisational tool in building workflows, permitting a workflow designer to aggregate portions of a workflow and create conceptual abstractions. In this usage, the composite actor does not contain a director, and is called a *transparent composite actor*. The hierarchy has no semantic consequences; it is just a syntactic device. A

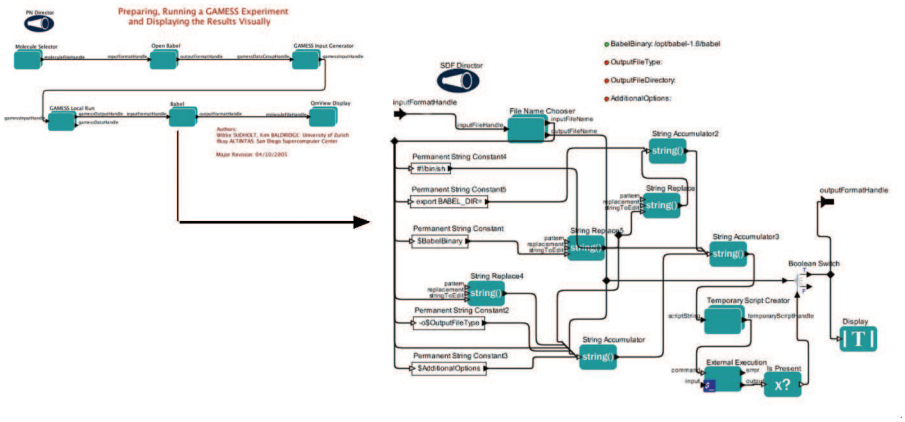


Figure 7.1: A Kepler workflow from chemistry combining the PN and SDF director.

second use of hierarchy is to use a workflow to define an actor. The Babel example is of this type. The fact that it has a director makes it function within the top level workflow exactly as if it were an *atomic actor*. A composite actor that contains a director is called an *opaque composite actor*, because its internal structure is neither visible nor relevant to the outside director.

For an opaque composite actor to function externally as if it were an ordinary actor, the director must be able to execute the inside workflow in a manner that emulates an actor. We examine below what that means, but before we can do that, we explain a few of the MoCs in enough detail that they can serve as illustrative examples.

7.4 Models of Computation in Ptolemy II and Kepler

One of the main objectives of the Ptolemy Project has been the exploration of models of computation. For this reason, many distinct directors have been created by various researchers, some realizing fairly mature and well-understood models of computation, and some that are much more experimental. Kepler has adopted Ptolemy's rich MoC architecture and focused principally on a few of the more mature ones, described here.

Process Networks (PN). In PN, each actor executes in a Java thread, and all actors execute concurrently. An actor can read input data encapsulated in tokens from input ports, and write data encapsulated in tokens to output ports. Normally, when it reads from an input port, the read blocks until an input token is available. Writes do not block. The PN director includes sophisticated scheduling policies to ensure that buffers

for tokens remain bounded, and also detects deadlock, which is where all actors are blocked attempting to read data. See [KM77] and [PGM97]. Most of the scientific workflows (composite actors) built with Kepler to date have been based on PN.

Dataflow (DDF and SDF). In dataflow MoCs, instead of having a thread associated with each actor, the director "fires" actors when input tokens are available to them. We discuss two variants of dataflow here, dynamic dataflow (DDF) and synchronous dataflow (SDF). In the case of DDF, the director dynamically decides which actor to fire next, and hence constructs the firing schedule dynamically at run time. In the case of SDF, the director uses static information about the actor to construct a schedule of firings before the workflow is executed, and then repeatedly executes the schedule. SDF is very efficient in that very little decision making is made at run time. PN is semantically a superset of DDF, in that the repeated firings of an actor in DDF can be viewed as (or even implemented as) a thread. Every DDF workflow can be executed using a PN director. DDF in turn is a superset of SDF, in that every SDF workflow can be executed identically with a DDF director. In SDF, a fixed number of tokens are consumed and produced in each firing. The token consumption and production rates allow for the computation of a fixed schedule. In SDF, deadlock and boundedness of communication buffers are decidable. As a consequence, SDF is well suited to code generation (the synthesis of a stand-alone program that executes independently of the workflow framework) [GZL07]. With DDF, actors need not have a fixed token production or consumption rate, the schedule is determined at runtime. In DDF, deadlock and boundedness are not decidable. In a DDF model, an actor has a set of firing rules (patterns) and the actor is fired if one of the firing rules forms a prefix of unconsumed tokens at the actor's input ports.

Continuous Time (CT). In CT, the communication between actors is (conceptually) via continuous-time signals (signals defined everywhere on a time line). The CT director includes a numerical solver for ordinary differential equations (ODEs). A typical actor used in CT is an *Integrator*, whose output is the integral from zero to the current time of the input signal. The CT director advances time in discrete steps that are small enough to ensure accurate approximations to "true" continuous-time behavior.

Discrete Events (DE). In DE, tokens communicated between actors are associated with a time stamp, a numerical value that is interpreted as the time at which the communication occurs. The DE director "fires" an actor when one or more of its input ports has the "oldest" (least time stamp) token among all the unconsumed tokens, or when the actor has requested a firing at a time stamp that is less than that of all unconsumed

tokens and all other pending requests for firing. When the actor fires, it consumes the input tokens, if any, and possibly produces output tokens. It may also request of the director a firing at some future time stamp.

Synchronous/Reactive (SR). In SR, every actor is (conceptually) fired on every "tick" of a global "clock." On each firing, an actor may observe input values and assert output values, but in any tick, even if it is repeatedly fired, if the inputs remain the same, then the asserted outputs should remain the same. The SR director fires all actors in every tick of the global clock repeatedly until all signals at all ports are defined. A signal is defined either if it has a token as its value or if it has been asserted to be "absent" (to have no token).

Finite State Machines (FSM) and Modal Models. An FSM composite actor is very different from the above. The components in an FSM composite actor are not actors, but rather are states. The FSM director starts with an initial state. If that state has a refinement, then the FSM director "fires" that refinement. It then evaluates guards on all outgoing transitions, and if a guard evaluates to true, then it takes the transition, making the destination state of the transition the new current state. A state machine where the states have refinements is called a Modal Model. A *Modal Model* is an opaque composite actor containing an FSM, each state of which may contain an opaque composite actor. In a modal model, the refinement of the current state defines the current behavior of the state machine. The refinement of a state need not have the same type of director as the workflow containing the modal model. When FSM is combined hierarchically with CT, the resulting models are called *hybrid systems.*

There are many other MoCs implemented in Ptolemy II, but the above set is sufficient to illustrate our key points. Akin to choosing between programming languages to tackle a problem, often different directors can be chosen to model a given phenomenon. A suitable director does not impose unnecessary constraints, and at the same time is constrained enough to result in useful derived properties (such as efficient execution or deadlock detection). The misinformed use of directors also leads to actors that cannot be embedded in others, as explained in the next section.

7.5 Composing Models of Computation

MoC composition is being explored in multiple scientific workflow systems. Examples other than Kepler include the Taverna and Inforsense systems. Taverna 2 allows computational steering of its data flows and within those data flows it uses FSM semantics

to manage policies for individual services [Oin07]. Inforsense too combine FSMs with dataflows [CGWG07]. Bar modal models however, MoC compositions have not been well treated in earlier research. Although prior work has offered formalisms for describing MoCs, e.g., [MB07] [HB07] and comparing them, e.g., [LSV98], a study of MoC compositions is lacking. To address the void, we develop a classification of valid MoC combinations in Kepler/Ptolemy II.

In the Kepler environment, opaque composite actors can be put into workflows with a different type of director, thereby combining different models of computation in one workflow. In the workflow in Figure 7.1, the Babel actor is part of a network of actors orchestrated by the PN director. The Babel actor internally uses an SDF director. In the example, SDF is nested inside PN, which is a valid combination, as we will explain below. Nesting PN inside of SDF would have been invalid in most cases. The choice of director determines whether a given actor can be put on the inside or outside of other actors.

To determine which combinations are possible, we need to know two things about a director:

1. What properties it assumes of the actors under its control, and
2. What properties it exports via the opaque composite actor in which it is placed.

If a director's exported properties match those assumed by another director, then it can be used within that other director. Otherwise, it cannot. In the example of Figure 7.1, the SDF director exports properties that match those assumed by the PN director, and hence SDF can be used inside PN. The properties in question can be formulated in terms of actor abstract semantics and director abstractions of time.

7.5.1 Actor Abstract Semantics

All models of computation in Kepler and Ptolemy II share a common abstraction that we call the *actor abstract semantics*. Actors and directors are instances of Java classes that implement the Executable interface, which defines *action methods*. The action methods include two distinct initialisation methods:

1. preinitialize(): invoked prior to any static analysis performed on the workflow (such as scheduling, type inference, checking for deadlock, etc.).

2. initialize(): invoked to initialise an actor or director to its initial conditions. This is invoked after all static analysis has been performed, but it can also be invoked during execution to reinitialise an actor.

The action methods also include three distinct execution methods that are invoked

in sequence repeatedly during an execution of the workflow:

3. prefire(): invoked to check whether an actor is ready to fire (for example, an actor may return false if there are not enough input data tokens).
4. fire(): In this method, the actor should read input tokens from input ports and write tokens to output ports, but it should not change its state. That is, if the fire() method is invoked repeatedly with the same input tokens, then the resulting output tokens should be the same.
5. postfire(): In this method, the actor can read input tokens and update its state.

Finally, there is a finalisation method:

6. wrapup(): invoked for each actor just prior to finishing execution of a workflow.

All of the methods are required to be finite (they must eventually return).

The method definitions specify a contract, but not all actors obey this contract. Any actor that strictly conforms to this contract is said to be *domain polymorphic*, and the actor may be used by any director that operates on actors (which is all the directors above except FSM, which operates on states).

Actors that do not obey the contract are more specialised, and may only work with specific directors. They are not domain polymorphic (strictly obeying the actor abstract semantics) and come in two flavours. The first flavour obeys a looser version of the abstract semantics where the fire() method provides no assurance that the state of the actor is unchanged. The second is still looser in that it that also provides no assurance that any of these methods is finite. Based on these three levels of conformance to actor abstract semantics, we can now classify the directors.

7.5.2 Abstract semantics assumed by a director of the actors under its control

The *PN* director only assumes the loosest of these abstract semantics. It does not require that any method be finite because it invokes all of these methods, in order, in a thread that belongs entirely to the actor. If an actor chooses to run forever in the preinitialize() method, that does not create any problems for the director. The director will let it run. *Dataflow* and *DE*[1] directors require that actors conform with the loose actor semantics, where all methods are finite. But they do not require that actors leave the state unchanged in the fire() method. *CT* and *SR* require that actors obey the strictest form of the semantics. The director iteratively fires actors until some

[1] A variant of DE is described in [9] that requires and exports strict semantics, but that is not what is implemented in the current version of the software (version 6.0 of Ptolemy II).

condition is satisfied. The strict actor semantics ensures that the answers will always be the same given the same inputs. *FSM* requires loose actor semantics. A firing of an FSM in Ptolemy II consists of a firing of the refinement of the current state (if there is one), followed by evaluation of the guards and a state transition. Clearly, the firing of the refinement must be finite for this to be useful.

7.5.3 Abstract semantics exported by a director via the actor in which it is placed

A director also implements the *Executable* interface. If a director conforms to the strict actor semantics, then an opaque composite actor containing that director also conforms to the contract. Such an actor can be used safely within any workflow. In the current version of Ptolemy II (version 6.0), only the *SR* director conforms to the strict actor semantics, although in principle *CT* and *DE* can be made to conform. Currently these and the *dataflow* directors conform to the looser abstract semantics, but still guarantee that all methods return after finite time. *PN* only conforms to the loosest version, providing no guarantees about methods ever returning. The *FSM* director exports whatever the state refinements export.

7.5.4 Abstractions of time

Some of the directors (notably *CT* and *DE*) explicitly manage a notion of the advancement of time. An actor, when it fires, can ask the director for "current time," and actors can expect that time will advance monotonically between firings. Other directors (notably the dataflow directors and *SR*) are agnostic about time. They will pass requests for current time up the hierarchy to the next director above them. If they are at the top level, then by default, they do not advance time, and hence time does not progress beyond a starting point (typically 0.0). However, the SDF and SR directors have a parameter that can be used to increment time between iterations of the model, thus providing a model of discrete, regular advancement of time. Some directors (notably *PN*), have no notion of time and no notion of an iteration, and hence cannot meaningfully advance time.

This model of time, along with the abstract semantics, imposes some constraints on the combinations of directors that can be used. Specifically, some directors require that time advances (*CT* and *DE*). These directors cannot be put inside a director that does not advance time (*PN*).

Inner director ↓ (**exports X**)	Outer director ↓ (*requires Y*)						
	PN (*loosest*)	SDF (*loose*)	DDF (*loose*)	CT (*strict*)	DE (*loose*)	SR (*strict*)	FSM (*loose*)
PN (**loosest**)	Yes	No	No	No	No	No	No
SDF (**loose**)	Yes	Yes	Yes	No	Yes	No	Yes
DDF (**loose**)	Yes	Yes	Yes	No	Yes	No	Yes
CT (**loose**)	No	Yes	Yes	No	Yes	Yes	Yes
DE (**loose**)	No	Yes	Yes	Yes	Yes	No	Yes
SR (**strict**)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
FSM (**refinement**)	Yes if the refinement is stricter than or equal to *Y*						

Table 7.1: Rules for hierarchically mixing directors in Kepler and Ptolemy II.

7.5.5 Director compatibility

We classify directors according to the following criteria:

1. They require that the actors they control are *strict*, *looser*, or *loosest*, depending on whether they must conform to the strictest, looser, or loosest form of abstract semantics.
2. They similarly export *strict*, *looser*, or *loosest*. Ideally, any director should export the same version of the contract it assumes or a stricter version, but this is not the case in the current version of Ptolemy II.
3. In addition, they either require time to advance, or do not require it.

The current status of the directors is given in Table 7.1. The rules applied to determine director compatibility are: (i) exported abstract semantics should be stricter than or equal to required abstract semantics and (ii) a director that requires that time advances should only be put inside a director that advances time. The table is on-line and will evolve (see `www.ptolemy.org/heterogeneousMoCs`).

7.6 Composing PN, Dataflow and FSM Directors

A key question may arise at this point. If actors can be made domain polymorphic by conforming to the strict actor semantics, then why not design all directors to conform? In some cases, the semantics of the MoC precludes this. In other cases, it would simply be too costly. We examine some of these cases.

PN. The PN director is apparently the least restrictive in the actors it can manage,

but also the least useful in an opaque composite actor. The reason for this is very fundamental. If the PN director were to define a finite fire() method, what should that method do? Each of the actors under its control is executing in its own thread of control. How much execution should be performed? One possible answer is "as little as possible," but this would result in nondeterminate execution. If two actors have threads that are able to perform computation, which should be allowed to perform computation? The only other obvious answer is "as much as possible." This can be made determinate, but typical PN workflows can execute forever if given the opportunity. Hence, this yields an infinite execution. PN is sufficiently expressive that determining whether this execution is infinite is equivalent to solving the famous halting problem in computation, and hence is undecidable. This property of PN explains why it is challenging to introduce a model of time to PN. Specifically, timed variants of PN that have appeared in the literature allow time to advance when the model deadlocks. But whether the model deadlocks is undecidable, and typical models never deadlock. Thus, the loosest abstract semantics comes at a serious price, an inability to introduce a model of time.

For example, the workflow of Figure 7.1 with PN on the outside would be hard to re-use inside others. It follows that, when a workflow has potential to be re-used inside others, PN should be avoided and, if possible, replaced by a more reusable director. Moreover, models that require time to advance cannot be used within PN.

DDF. DDF is as expressive as PN, and hence potentially suffers from the same limitation. However, DDF has an advantage. It assumes that all actors under its control have finite firings. Thus, it is relatively easy for the designer of a workflow to specify how many firings of the component actors constitute a single firing of the enclosing opaque composite actor. The DDF director assumes a simple default if these numbers are not given by the workflow designer: one firing of a DDF opaque composite actor constitutes at most one firing of each component actor. The actor is fired if possible, and not fired if not, given the available input data. This yields a simple, finite, and determinate notion of a finite firing for the director to export. This enables the introduction of a model of time. Time can advance between firings of the director. If the DDF director is used at the top level, the amount of the time advance would need to be given by a parameter. If it is inside a timed domain like DE, then the amount of the time advance can be specified by the environment.

SDF. SDF is still simpler in that it is not as expressive as PN, and there is a simple unique finite firing that is natural and easy to define. However, for both DDF and SDF, it is difficult to define a fire() of an opaque composite actor that does not update the state

of the workflow because data values stored on buffers change during the firing of the component actors. In order for SDF and DDF to export the strict actor semantics, they would have to backtrack or restore the state of these buffers on repeated invocations of the fire() method.

FSM. A particularly interesting case is FSM and modal models. Modal models always use opaque composite actors as state refinements, and these must at a minimum have finite firings to be useful (given the semantics of an FSM in Ptolemy II discussed before). Hence, it does not make sense to use PN inside the refinement of a state. But any other of the directors described above can be used in a Modal Model.

Figure 7.2 illustrates the differences with a small example that shows how DDF and modal models can be embedded in SDF. The model starts with a Ramp actor that produces tokens starting at 0 and then increasing by 1. The signal is then distributed to a DDF opaque composite actor and to a modal model, both of which randomly change the gain of the signal. The DDF portion uses conditional routing of the tokens to route each token through one of two actors that either multiplies the value by 1 or by -1. In a similar fashion, the modal model has two refinements that either multiplies the value by 1 or by -1. The output is then plotted; see Figure 7.3. The refinements of the modal model are SDF models. Thus, the modal model refinements export SDF semantics and could be embedded inside PN if PN was at the top level. Swapping the DDF director with PN makes the workflow fail because PN cannot be used inside the outermost SDF. The reason is that PN conforms to the loosest execution semantics and makes no guarantee about ever returning. This example is available on-line for exploration via the above mentioned Web site.

7.7 Composing SR, DE, and CT Directors

SR and CT both require and export the strict actor semantics. DE has the potential to do so, although the current software implementation is looser. This fact creates an interesting suite of modeling capabilities for timed systems, that when combined with FSM, is extremely useful. In particular, it is possible for SR, DE, CT, and FSM to be combined hierarchically in any order. More interestingly, as shown in [LZ07], DE can be thought of as a generalisation of SR, and CT as a generalization of DE. Each generalisation adds expressiveness at the expense of efficiency. There are also subtle stylistic differences between these directors that make them all useful, despite being generalisations of one another. Details and examples are given in [LZ07].

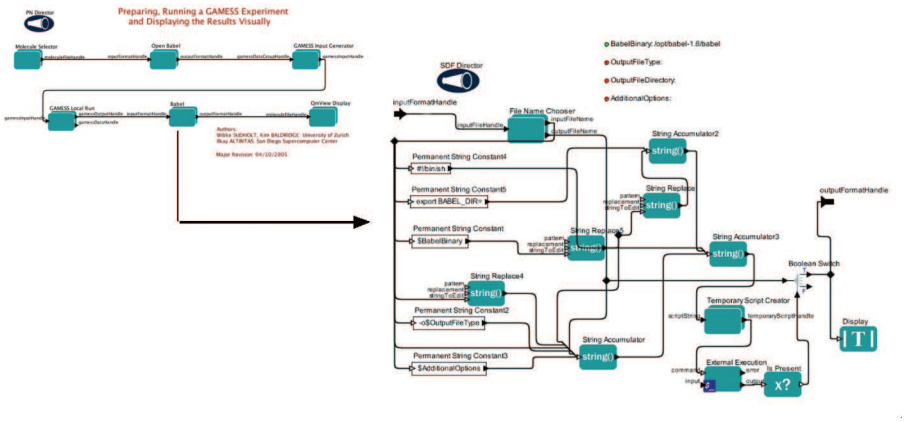


Figure 7.2: A simple example with DDF and FSM directors inside an SDF director.

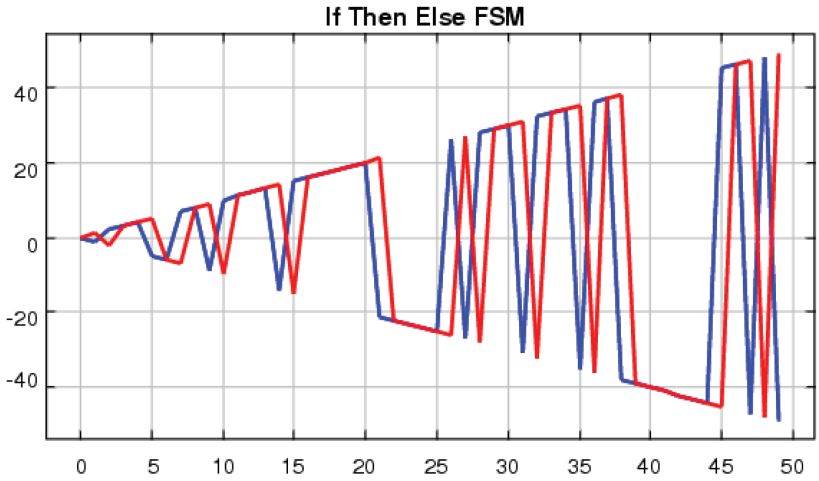


Figure 7.3: The output of the example MoC composition.

7.8 Summary and discussion.

There are scenarios in e-science that rely on composing models of composition. Workflow re-use in the presence of multiple models of computation is non trivial. Based on the notion of hierarchy and actor abstract semantics, we developed a classification of models of computation available in the workflow environments Kepler and Ptolemy II. The classification shows the compositions that are possible and useful. It turns out that some models of computation are more re-usable than others. Notwithstanding several restrictions, many compositions are possible. Time-based simulations can be mixed with dataflow and finite state machines can be combined with almost anything.

Our exploration of compositions of models of computation should be relevant for scientific workflow systems wishing to add multiple models of computations to their modelling capability.

The analysis is also relevant to help predict the behaviour of workflow systems which implement different models of computation and are combined to run integrated experiments. The work raises interesting questions on what is Kepler/Ptolemy specific and how one could extend the approach to formulate the abstract semantics of any workflow engine. For example, what is the impact of using the actor paradigm? If all workflow systems could be described in terms of their abstract semantics, a table detailing how multiple workflow systems are to be combined becomes an option. This is highly relevant for workflow sharing, execution and interoperation platforms such as `www.myExperiment.org` where multiple workflow engines run concurrently.

Chapter 8

Conclusions

The goal of this thesis is to investigate workflow re-use and discovery in science. To achieve this goal we investigated the following four research questions:

Q1 What requirements should be fulfilled for workflow re-use and discovery in science to occur?

Q2 How do scientists re-use and discover workflows?

Q3 Can automated discovery techniques support workflow re-use and discovery?

Q4 How does having multiple models of computation in a workflow affect workflow re-use?

8.1 Conclusions and Contributions

This section provides an overview of our conclusions and contributions in terms of the four research questions. The overall conclusion of the work is that scientific workflow re-use is a novel problem with its own set of requirements (see Section 8.1.1) which are partially addressed by automated workflow discovery techniques. We found that it is possible to capture how one class of scientists, bioinformaticians, approach the re-use and discovery of data flow oriented workflows (Section 8.1.2). This outcome provided a benchmark enabling the evaluation of workflow discovery techniques in the bioinformatics domain, to positive effect (discussed in Section 8.1.3). Finally, when considering scientific domains other than bioinformatics, we observed that scientific problems are modelled in multiple models of computation. We analysed the impact on

workflow re-use of having workflows with multiple models of computation (Section 8.1.4).

8.1.1 Requirements for workflow re-use and discovery in science

The concept of a workflow is becoming commonplace in large-scale science. Chapter 2 provided an overview of the scientific workflow landscape by introducing requirements from different application domains and by contrasting different workflow system environments. Multiple domains have found that a data flow model of computation is the right paradigm to model computational analyses in. In the thesis, we took the same view and adopted a formal definition of a workflow as a data flow.

With the rise in available scientific workflows comes the potential for their re-use. Workflow re-use is a subclass of software re-use, which is a well-researched yet tenacious problem in software engineering. To clarify the specifics of the problem in the science context, we elicited requirements from scientists and developers of workflow systems in science.

Requirements for workflow re-use

Chapter 2 highlighted several cases of workflow re-use in distinct scientific domains, including bioinformatics, earth sciences and ecology. Based on the results of three user surveys in Chapter 2, we claim that workflow re-use in science only happens when the following three categories of requirements are met:

The availability of re-usable workflows. A sharing infrastructure should be in place to make workflows available. Such workflows would need to be re-usable. Workflow re-usability is affected by the choice of model of computation (see Section 8.1.4), workflow language, abstractions used in the workflow environment system, the use of data models and the use of distributed autonomous services.

A community open to workflow re-use. Sharing attitude, re-use skillset and re-use motivation drive the dynamics of the workflow re-use cycle. Different communities are marked by different dynamics.

Effective and efficient workflow discovery. Determining the relevance of a workflow to a given problem is challenging. Being able to process the available number of workflows, having enough quality workflow documentation to assess relevance

and an effective ranking of potential solutions are key requirements for finding relevant workflows.

Requirements for workflow discovery

Effective and efficient workflow discovery is a prerequisite for workflow re-use. Workflow discovery occurs at different times in a workflow's lifecycle: (i) while the workflow is still being designed, (ii) post design, as a finished, concrete workflow, (iii) post design, as a finished yet abstract template which still needs to be completed dynamically during enactment, (iv) during enactment, including intermediary results and (v) post enactment, including results. We focussed on discovery of finished, concrete workflow. We elicited specific requirements from scientists and developers in Chapter 3. A number of discovery needs emerged, which lead us to the following characterization of workflow discovery.

A definition of workflow discovery tasks

To characterise the different discovery scenarios, workflow discovery tasks were identified, differentiating on three main axes:

Supporting re-use versus repurposing. Concrete workflow discovery is a sub-task of both workflow re-use and repurposing. When supporting re-use, workflow discovery shows similarities to traditional service discovery. When supporting repurposing however, we claim that workflow discovery is dissimilar to service discovery. In this case it concerns "composition-oriented discovery," which sits in between service discovery and service composition.

User query context. The provided user input to a workflow discovery problem varies from the workflow a user is currently working on, a textual statement of the workflow one wants, a stated research hypothesis to a document with the results of a brainstorming session.

Similarity versus complementary. Both when used to support re-use or repurposing, discovery tasks involve establishing how workflows are similar or complementary to the user query context.

This led to *a formal definition of workflow discovery tasks*. We defined the discovery tasks formally for the case of querying workflows by example. In a query by

example approach to discovering workflows, one queries for workflows "by example," taking in an existing workflow representation in order to find others. A query by example approach is feasible for any user query context as long the user input can be turned into a workflow-like representation.

The discovery tasks were formalised by relying on two novel types of *workflow conditions*:

Workflow similarity conditions. They define similarity between two workflows' constituent services and their data links.

Workflow complementarity conditions. They define complementarity between two workflows' services and tree branches.

The similarity conditions are relevant when discovering workflows that are similar to the user query input. This is the case when users are (i) simply re-using workflows and (ii) during repurposing, when they are looking to make workflow replacements. The complementarity conditions are relevant when discovering workflows that are complementary to the user query input. This is the case during repurposing when users are (i) looking to make workflows extensions and (ii) looking to make insertions. Combinations of both types of metrics are relevant when overlaps between matching workflows are considered.

8.1.2 Capturing workflow re-use and discovery by scientists

In the controlled environment of a series of user experiments, Chapter 4 explored how one class of scientists, bioinformaticians, do workflow re-use and discovery. The following findings were obtained.

Workflow re-use and discovery requirements confirmed

The *relative impact of several of the earlier posited requirements on re-use and discovery* were tested. Table 8.1 summarises the outcomes. We altered the following parameters between the experiments:

The discovery task expected of participants. The discovery task involved either finding similar workflows or complementary workflows. In the last experiment, we also recorded how participants edited the workflows found.

The relation of the participant with respect to workflow authorship. She is either discovering her own workflow, re-using one by a collaborator or one by an external party.

Participant motivation to complete the exercises. This ranged ranged from low to high in the experiments, depending on the experimental setup.

The expertise of participants. Low expertise means participants had almost no experience with bioinformatics. Medium means their main background is in another field but they understand basic bioinformatics notions. High means they are active in bioinformatics research.

The quality and amount of workflow information available per workflow. Low quality and amount means only a workflow's diagram is shown with the names given by the original author. High quality and amount means a professional curator added natural language text explanations and semantic concepts to the workflow and its constituent services.

The outcome of each experiment was judged to be positive only when the results from the exercises showed a level of agreement between participants and were confirmed by a bioinformatician as being sensible. The experiments revealed the following:

- *Bioinformaticians are capable of all types of workflow discovery* we considered, when the conditions are right.
- The commonsense expectation is that participant familiarity with the workflow author, participant motivation and participant expertise correlate positively with valid answers to discovery tasks. This expectation was confirmed in all experiments.
- Lots of quality workflow documentation is no requirement to achieve good results when it comes to discovery of one's own workflows or workflows by collaborators, as shown by experiments 3 and 4.
- Experiment 5 showed that the combination of motivation, expertise and quality metadata enables discovery from external parties. Contrasting this finding with the outcome of experiment 2 where participant expertise was medium suggests that either the motivation factor or the quality documentation factor could be the

Exp.	Discovery task	Relationship with original workflow author	Motivation	Expertise	Documentation quality and amount	Outcome
1	Find similar	External parties	Low	Low	Low	Negative
2	Find similar	External parties	Low	Medium	Low	Negative
3	Find similar	Original author	High	High	Low	Positive
4	Find similar	Collaborators	High	High	Low	Positive
5	Find and edit similar and complementary	Collaborators and external parties	High	High	High	Positive

Table 8.1: Summary of user experiments into workflow discovery.

deciding one. Taking into account that the motivation in experiments 1 and 2 was affected substantially by the lack of documention, we concluded that *documentation plays a crucial role* either indirectly (to drive motivation) or directly (to inform the discovery process).

A better understanding of workflow re-use and discovery behaviour

Experiments 1 and 5 were designed to create an understanding of the behaviour bioinformaticians exhibit during workflow re-use and discovery. Experiment 1 formulated *a range of plausible hypotheses* to uncover patterns of searching and matching bioinformaticians use to establish workflow similarity in general. How do they rank workflows? What criteria are used and in which combination? However, it turned out the task as presented to inexperienced bioinformaticians was too difficult and no conclusive results could be drawn.

Experiment 5 also had the ambition to model bioinformatician behaviour, yet focussed less on the notion of the similarity and more on *how the overall re-use process is performed*, in particular the *assessment of relevance of potential workflow candidate*

Benchmark	Experiment	Number of participants	Behaviour captured	Number of assessments	Participant agreement (Kappa value)
1	3	2	Similarity assessments	145	N/A
2	4	2	Similarity assessments	456	Very good (0.678)
3	5	24	Relevance assessments	1848	Very good (0.666)
4	5	24	Editing	Unprocessed	Unprocessed

Table 8.2: Summary of human benchmarks for workflow re-use

solutions and their subsequent editing. It presented specific re-use cases to experienced bioinformaticians.

The same study showed that *relevance assessment and editing are done in two distinct phases*. First, participants scanned the whole population of available workflows. After this, editing was done on the workflows marked as relevant.

It also documented *which sources of information are used in which phase*. For both phases, the workflow diagram was the first and most used point of recourse for finding information, despite its low detail and ambiguity. This finding underlines the power of using a visual medium. Textual workflow and service inputs and outputs were also used eagerly in both phases, but less so than the diagram. The overall workflow description and workflow name were deemed useful for relevance assessment only.

A suite of benchmarks for workflow re-use and discovery

The work of participants translates into a documented set of decisions made during the workflow re-use process. The three user experiments which had positive outcomes contribute four benchmarks with different characteristics. They are available from `www.myexperiment.org/benchmarks`.

Benchmark 1 collects similarity assessments made by a workflow author about pairs of her own workflows. In Benchmark 2, a collaborator made similarity assessments on those same workflows. Benchmarks 3 and 4 rely on the re-use tasks solved by participants in user experiment 5. Benchmark 3 contains the assessments made regarding the relevance of a set of workflows to solve the re-use task in question. Benchmark 4 captures the next step in solving the task by collecting the edit operations participants

undertook to solve it. Due to time constraints, these edit operations have not been entered into electronic format yet and their statistical analysis has not been performed.

All benchmarks were created by participants who felt confident while creating them. For benchmarks 2 and 3, they also agreed strongly on the assessments made, as shown by the Kappa statistic for inter-rater agreement. Even though a high level of agreement was reached, agreement was never perfect. For the case of benchmark 3, the disagreement could be measured in terms of correctness of answers. Contrasting relevance assessments with the correct solution, participants on average were right in 73% to 91% of cases, depending on the scheme used to assess a given answer. Analysis of the data uncovered the following main sources of disagreement: (i) documentation incompleteness or ambiguity, particularly involving data types, (ii) assumptions made about the required generality of a solution and (iii) assumptions made about the admissibility of additional, external "shim" or glue services to create a solution.

8.1.3 Supporting workflow discovery with automated techniques

The user experiments showed that workflow re-use and discovery is difficult for bioinformaticians. To support them, in Chapter 5 we considered the use of a range of automated discovery techniques. We evaluated the worth of the techniques by comparing their results with those of the participants on tasks presented during the user experiments.

Automated discovery techniques

The investigated techniques were built to support one type of workflow discovery task. They differ mainly on the type of information they exploit.

1) Type of workflow discovery tasks supported. All the techniques considered work based on similarity-based matching of workflows. None were designed explicitly to support complementarity-based matching between workflows.

2) Type of workflow documentation exploited. Given the expense of obtaining high quality workflow documentation, we considered techniques which operate on different levels of documentation. Table 8.3 organises the techniques by the formalism they rely on to query workflows and by the level of detail at which workflows are queried. Text refers to techniques using natural language descriptions. RDF refers to techniques using light-weight semantic descriptions. OWL refers to techniques using rich semantic descriptions. Other resources can equally be described in natural language, RDF or

Formalism	Technique not using workflow structure	Technique using workflow structure
Text	Google API, Woogle4WF	GUB4WF
RDF	JMFeta	
OWL		OWL4WF

Table 8.3: Summary of the considered automated discovery techniques.

OWL. The fact that workflows are involved means the notion of structure is central – where possible, the relationship between constituting services needs to be retained.

Using natural language descriptions. Three techniques were explored to exploit textual information in workflows. Firstly, a simple wrapper to the Google search engine API was built to investigate Google's performance on unprocessed workflow descriptions. Secondly, we adopted the existing Woogle service discovery tool for text-based service discovery (unrelated to Google) and adapted it into the Woogle4WF tool which translates a workflow to look like a service. Thirdly, to enable querying the internal structure of a workflow in conjunction with textual descriptions of its constituent services, we developed GUB4WF, a tool exploiting an existing graph matching library for sub-isomorphism detection.

Using light-weight semantic descriptions. We adopted the existing JMFeta tool for semantic service discovery. The tool relies on a similarity metric and annotations of services made based on the RDF(S) subset of the Resource Description Framework (RDF) language. It allows to query for similar workflows by providing it with workflow annotations that look like regular service annotations.

Using rich semantic descriptions. We analysed the possibilities and limitations of using the description logic based Web Ontology Language, OWL, for querying the structure of data flows and proposed a novel *workflow ontology*. The combination of using selected example queries over the developed workflow ontology with the Racer description logic reasoner was dubbed OWL4WF.

Evaluation of automated discovery techniques

We evaluated the performance of the techniques by comparing their solutions on re-use tasks with the solutions provided by bioinformaticians. Several choices were made as

Data set	Discovery task	Google API	Woogle4WF	GUB4WF	JMFeta	OWL4WF
Experiment 1	Similarity - cross author			Case		
Benchmark 1	Similarity - personal		Full	Full		
Benchmark 2	Similarity - cross author	Case	Full	Full		
Benchmark 3	Similarity and complement - cross author				Case	
None	Similarity - personal and cross author					Case

Table 8.4: Summary of evaluation method for automated discovery techniques

part of the evaluation method. After recalling these, we summarise the major findings of Chapter 6.

1) Evaluation method choices. We made several *pragmatic choices in performing the evaluation*. Ideally, one would measure the performance of all techniques by comparing them against the solutions of participants on all performed workflow discovery tasks for those user experiments which yielded statistically coherent results. In practice, we evaluated only particular combinations of techniques and data sets. Table 8.4 lists these combinations. Although the table looks sparse, it covers a wide range of techniques and tasks.

The techniques were either (i) fully evaluated against the assessments of all participants on all tasks in a data set, (ii) evaluated partially against the assessments of all participants for the case of an individual task, or (iii) not evaluated on a particular data set. The lack of full evaluation was due to three reasons: unavailability of annotation, predicted performance and time constraints in processing benchmark results.

In addition, in one case we relied on a statistically unconvincing data set. The potential and limitations for producing workflow rankings based on the GUB4WF tool were partially demonstrated based on data from user experiment 1.

2) Findings. Keeping these considerations in mind, we reached the following overall conclusions:

Finding 1. *Workflow discovery techniques are task specific.* All the data set based evaluations provide evidence that the discovery techniques under consideration are

highly task specific. This suggests that, *to support the full range of discovery tasks, a suite of techniques will be needed.* The results for the techniques on the different discovery tasks are summarised as follows:

Similarity-based personal discovery. The use of a graph matcher in GUB4WF to represent workflow structure proved valuable to support personal discovery (GUB4WF + Benchmark1; GUB4WF + Experiment1). Conversely, the lack of a structural matching component in the Woogle4WF tool meant it was unable to detect any structural relations between workflows which resulted in poor precision (Woogle4WF + Benchmark1).

Similarity-based cross-author discovery. The use of a text clustering component in Woogle4WF proved valuable to support cross-author discovery (Woogle4WF + Benchmark2; Woogle4WF + Benchmark3). Conversely, the lack of a text clustering component in GUB4WF meant it was unable to detect results during cross-author discovery (GUB4WF + Benchmark2; GUB4WF + Experiment1). Finally, the combination of the Google API based wrapper and the proprietary ranking mechanism used by Google for unprocessed workflows yielded poor results for cross-author discovery (Google API + Benchmark2).

Complementarity-based discovery. The performance of applying light-weight semantic and text-based similarity-based techniques on complementarity-based tasks was poor (Woogle4WF + Benchmark3; JMFeta + Benchmark3). The results from the limited case study based on the JMFeta tool (JMFeta + Benchmark3) suggest that *the indiscriminate use of lightweight semantics offers little benefit over the indiscriminate use of the Woogle4WF text-based technique*.

Finding 2. *OWL reasoning enables a unique feature for workflow discovery but extensions to the underlying technology would make it even more useful.*

The open world semantics of OWL allow to *describe and query incomplete workflows*. This is a useful feature because (i) building workflows can take a very long time (ii) workflows can come annotated incompletely and (iii) workflows sometimes contain sensitive information and not all of the information in the workflow will be described for the outside world.

Extensions are needed to the description logic machinery underlying OWL. In particular, the querying of workflow fragments provides *a use case for including*

role composition capability to the current version (1.0). In addition, *the limited understanding and expressivity of current similarity-based retrieval techniques affects the usefulness of OWL to organise and retrieve similar workflow structures*.

8.1.4 Impact of multiple models of computation on workflow re-use

Our focus thus far has been on workflow re-use and discovery in the context of a single class of models of computation, namely data flows. However, in science, multiple models of computation are in use to model analyses. In this context, we studied the impact that having multiple models of computation has on workflow re-use. We collected use cases for workflows re-using workflows with alternative models of computation and analysed when these re-use cases are valid computationally.

There exist use cases for combining multiple models of computation inside a single workflow. Chapter 2 introduced a series of scenarios whereby the ability to combine heterogeneous models of computation within one workflow was crucial to modelling an experiment.

Workflow re-use in the presence of heterogeneous models of computation is often feasible. Section 8.1.1 stated that models of computation were one deciding factor for determining whether workflows were re-usable. Chapter 7 investigated the precise conditions under which workflows are re-usable in the presence of multiple models of computation. Notwithstanding several restrictions, we found that in many cases workflows using alternative models of computation can be re-used and embedded in a hierarchical manner. Time-based simulations can be mixed with data flows; discrete dynamics can be mixed with continuous dynamics; and finite state machines can be combined with almost anything.

Relevance to e-science middleware projects. The resulting classification should be relevant for scientific workflow systems wishing to add multiple models of computations to their modelling capability. The analysis is also relevant to help predict the behaviour of workflow systems which implement different models of computation and are combined to run integrated experiments.

8.2 Future work

The work in this thesis can be expanded in several directions. We revisit each of the research questions and set out possible avenues for future research. Future work with respect to the impact of multiple models of computation on workflow re-use is treated inside the requirements section.

8.2.1 Requirements for workflow re-use and discovery in science

Re-use across multiple workflow environments. This thesis focused on workflow re-use in the context of a single workflow environment. There is no standardised workflow language in the science community and it is unclear whether one will emerge. There is also a desire to maximize workflow re-use potential. The question whether to support workflow re-use across workflow languages is an issue for sharing environments like ^{my}Experiment.org where workflows stemming from different workflow systems are shared. What can be meaningfully re-used across workflow systems? Suppose I find a workflow created in an environment that is not my own - what can I meaningfully do with it? One option would be to integrate it in my workflow while executing it based on its original workflow engine. Examples of such integration exist between the Taverna [OGA+05] and VL-e [ZBB+06] environments. In case I would like to make changes to it, however, could I import it as is into my environment, or import only parts of it, or only its data products?

An ambitious research goal is to investigate workflow interoperability at the language level. Since the mid nineties, the Workflow Management Coalition (WfMC) has tried to address the lack of a standard between workflow systems in the business world. To stimulate interoperability between vendors, it has defined the Workflow Reference Model, shown in Figure 8.1, that outlines five key interfaces that a workflow management system must have. Since we are interested in workflow discovery, Interface 1 for defining the business process is highly relevant. It includes two aspects: a process definition expression language and a programmatic interface to transfer the process definition to/from the workflow management system. To define Interface 1, the XML Process Definition Language XPDL was introduced, a process design format for storing the visual diagram and process syntax of business process models, as well as extended product attributes.[1] XPDL is not an executable programming language like BPEL or Scufl, but rather a process design format that literally represents the drawing

[1] Web site: `http://www.wfmc.org/standards/xpdl.htm`

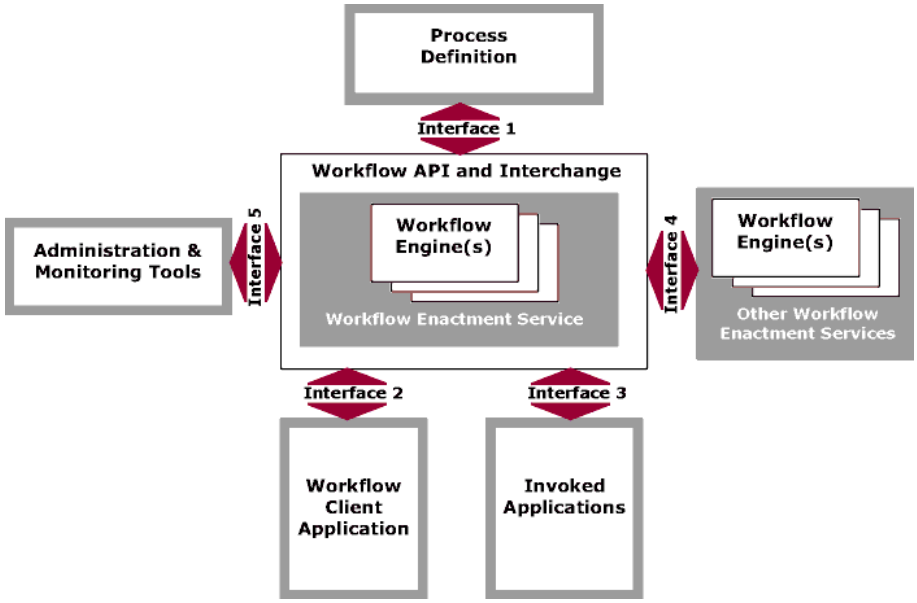


Figure 8.1: The Workflow Management Coalition Reference Model.

of the process definition. Specifically, it has vector coordinates, including lines and points that define process flows. As [Wen06] writes, the goal of XPDL is to store and exchange the process diagram. It allows one workflow design tool to write out the diagram, and another to read the diagram, and for the picture that you see to be as similar as possible. It does not, however, guarantee the precise execution semantics [Wen06]. We observed that research to identify, categorise and formally describe the different semantics of a workflow is ongoing and limited. Chapter 7 made an initial contribution. Further work is needed to chart the relationships between models of computations.

Discovery tasks given a variety of models of computation. Given the existence of multiple models of computation, is a universal representation and a universal set of requirements for workflow discovery feasible?

Several initiatives have focused on defining a common representation to improve workflow interoperability. For instance, the aforementioned XPDL was designed to support interoperability, not discovery. Similarly, the VL-e workflow interoperability bus proposal [ZBB+06] concentrates on interoperability, not on supporting discovery of concrete workflows.

The use of different models of computation implies new workflow discovery tasks and metrics could be defined, different to the ones we defined for data flows in this

thesis. Examples include measuring similarity and complementarity of workflows implemented as Finite State Machines or those implemented in a time-based model of computation. The representation required to support those tasks will be specific to the model of computation in question.

We expect however that some discovery tasks will be common across models of computation and that the representation required to support those tasks will be universal. For example, all workflows in a Web context will share the notion of a Web service. Comparing which services are shared and not shared by workflows is a powerful discriminator.

Finally, the possibility to combine workflows with different models of computation adds further complexity to deciding the level of similarity and complementarity between workflows.

8.2.2 Capturing workflow re-use and discovery by scientists

Workflow re-use and discovery requirements confirmed

The impact of high motivation and low expertise. In the user experiments, some ambiguity remains over the determining factor of certain outcomes (see Table 8.1 on page 185). In particular, it would make sense to measure the performance of participants who had little expertise with a domain but who were highly motivated. This case reflects the case of cross-disciplinary workflow re-use.

Crossing over to other disciplines. It would also make sense to organise the user experiments in scientific communities with similar and dissimilar re-use dynamics. This would provide evidence whether the developed model of re-use dynamics is generic enough and allows us to determine whether the conclusions drawn based on one domain translates to another domain.

A better understanding of workflow re-use and discovery behaviour

Factors driving human similarity search. The range of plausible hypotheses put forward in user experiment 1 to uncover patterns of searching and matching bioinformaticians use remains untested. The recording of such behaviour would help the design of future discovery tools. One useful addition to the existing hypotheses is to measure the relevance of the workflow similarity and complementarity metrics defined in Chapter 2 for human decision making. Another possible addition is to measure how participants deal with quality of service data in workflows.

Investigating navigation as search behaviour. Our approach towards discovery has largely ignored the navigation search paradigm. It is worthwhile to explore how a navigation based approach, using for example hypertext, compares with (or complements) a query-based tool.

Relevance assessment through execution. All user experiments consisted of paper-based exercises. Workflows however are programs and running a program makes people understand them better. One should not ignore the impact that trying to execute a workflow has on determining its relevance for a given re-use task.

Log-based analysis. As more workflows get discovered on workflow sharing sites such as myExperiment.org, analysis of such sites' log files provides a valuable resource to identify human workflow discovery behaviour.

A suite of benchmarks for workflow re-use and discovery

A generic benchmark for the scientific workflow community. Our developed suite of benchmarks captures re-use behaviour in data flows. Each scientific workflow system capable of modelling data flows should be able to re-model the workflows used into its own language.[2] It could then test its own discovery system with respect to the benchmarks. The fact that the workflows are from bioinformatics should not matter provided the discovery system in place is domain independent.

Guidance for the selection of workflow matching types We identified a number of workflow matching types. Which type is the most appropriate for a given workflow discovery task depends on the application domain. Analysis of specific re-use cases should yield heuristics to guide the choice. The collected benchmark could serve as the basis for such an analysis. This in turn could guide the design of discovery tools.

Analysis of edit operations. User experiment 5 collected the editing of workflows to solve the set tasks. The capture and analysis of the edit operations in this experiment should provide a useful resource to train and evaluate future techniques which take into account workflow structure and workflow complementarity.

[2] An exception are those services in the workflows that are of a type that is specific to the Taverna workbench.

8.2.3 Supporting workflow discovery with automated techniques

Automated discovery techniques

Deploying the considered techniques. None of the techniques considered in the thesis have been deployed in an operational setting. To deploy the GUB4WF graph matching based tool for cross-author discovery tasks, the addition of a lexical component would be essential. As for Woogle4WF text clustering based tool, it is brittle when it comes to spelling errors but it has proven a useful tool for finding similar workflows. The JMFeta semantic service discovery tool was not useful for finding complementary workflows. Further tests on Benchmark 3 should reveal whether JMFeta is more effective on retrieving similar workflows. The Google API based wrapper is likely to be more useful if workflows were published as rendered Web pages and not just unprocessed files. One possibility would be to publish them on a place like the myExperiment.org site.

Inexact graph matching. One promising area to look at for novel tool development is *inexact graph matching*. The Relaxation by Elimination graph matcher developed at the Advanced Computer Architecture Group at York and its commercial spin-off, Cybula, is one example of a robust, expressive and optimised inexact graph matcher. The addition of a lexical component to the matcher could prove a powerful combination to address both workflow similarity and complementarity based tasks over workflow corpora authored by different authors. The VisTrails system holds a similar promise. It implements an inexact graph matcher without a lexical component [SVK+07].

Generating workflow annotations. Techniques to address the knowledge acquisition bottleneck for workflows are needed. Techniques from service ontology learning and automated service annotation are promising in this respect. One could extend such work to address the identification and classification of workflow fragments, by taking into account the structure of fragments when applying the machine learning techniques this type of work relies on. Techniques from Web page usability mining also promise to assist in capturing the behaviour of scientists as they construct a workflow, make mistakes and then take corrective actions.

The power of the crowd. Web 2.0 style “tags” and social networks are a further source of workflow annotation. As scientists become more accustomed to on-line social networks and to tagging their workflows, such networks and tag clouds will become more relevant. How and at what level a workflow and its constituent services would be described, compared and their similarities aggregated are open questions.

Interaction between composition and discovery tools. Discovery tools working over a pool of existing workflows (i.e. detecting workflow extensions) co-exist with composition tools that predict potentially valid service compositions. The two classes of tools can (i) complement each others findings (for example when no particular service combination has ever been made before, whereas semantic information indicates that two services are compatible; or no semantic annotation is available but the workflow repository contains a particular service combination), or (ii) reinforce each others findings (for example when both approaches provide evidence for a particular workflow extension). What are the interaction models for these tools - which type of tools should be given more weight when?

Interaction between syntactic and semantic matching. Discovery relies on the notion of data, service and workflow fragment similarity (and complementarity). Similarity at the syntactic level does not imply semantic similarity. Conversely, similarity at the semantic level does not imply syntactic similarity. For example, at the data level, polynomous data sets may be referring to the same data set. At the service operation level, parameter names may use different terminology and structural types to indicate the same concept. At the workflow level, different compositions of services in different workflows can represent identical functionality. How do we manage and resolve the conflicts between the syntactic and semantic level all the way up the workflow stack, from data products to services to complete workflows?

Interaction between workflow lifecycle stages. The flexible interaction between workflow discovery techniques for concrete workflows and techniques for querying provenance logs, Web services and electronic lab books will facilitate the workflow design and re-use process. How can we find workflow designs based on wet lab protocols, mindmaps, provenance records? How can we find relevant provenance logs based on workflow designs (editing context), wet lab protocols, mindmaps?

Hybrid approaches. Workflow documentation for concrete workflows comes in different guises (identifiers, text, tags, ontology concepts, publications). Techniques that are able to combine the different sources of information associated with a single workflow are likely to be more robust and more effective. One potential way to approach the problem is through the lightweight combination of existing techniques and the introduction of a voting scheme on top to weigh the relative importance of the techniques. What are the interaction models for these tools - which type of tools should be given more weight when?

Evaluation of automated discovery techniques

Measuring subjective technique performance. For user experiment 5, the performance of the techniques in the eyes of scientists could be measured by having the participants rate the anonymised results of the techniques and compare them with their own anonymised results and those from other participants.

Scalability. An issue that was considered out of scope for the thesis was the responsiveness and accuracy of the considered tools in terms of the size of the repository of workflows.

Appendix A

List of Participants for User Experiment 5

1. (Anonymous), Lawrence Livermore National Laboratory, USA
2. Adam Barker, National eScience Centre, UK
3. Anika Joecker, Max-Planck Institute for Plant Breeding Research, Germany
4. Arnaud Kerhornou, European Bioinformatics Institute, UK
5. Bela Tiwari, NERC Environmental Bioinformatics Centre, CEH Oxford, UK
6. Ben Mahy, Vlaams Instituut voor Biotechnologie, Universiteit Gent, Belgium
7. Benjamin Good, iCAPTURE Centre, University of British Columbia, Canada
8. Cornelia Hedeler, School of Computer Science, University of Manchester, UK
9. Duncan Hull, School of Chemistry, University of Manchester, UK
10. Francois Moreews, SIGENAE team, INRA, France
11. Giovanni Dall'Olio, Grup de Recerca en Informatica Biomedica, IMIM, Spain
12. Hannah Tipney, University of Colorado Denver Health Sciences, USA
13. Helen Hulme, School of Computer Science, University of Manchester, UK
14. Lin Ching - Fong, National Yang-Ming University, Taiwan

15. Marco Roos, Instituut voor Informatica, Universiteit van Amsterdam, The Netherlands
16. Mark Wilkinson, iCAPTURE Centre, University of British Columbia, Canada
17. Mike Cornell, School of Computer Science, University of Manchester, UK
18. Nathan Nicely, Renaissance Computing Institute (UNC), USA
19. Paolo Romano, Bioinformatica, Istituto Nazionale per la Ricerca sul Cancro, Italy
20. Peter Li, School of Chemistry, University of Manchester, UK
21. Peter Rice, European Bioinformatics Institute, UK
22. Pieter Neerincx, Wageningen Universiteit en Researchcentrum, The Netherlands
23. Tim Booth, NERC Environmental Bioinformatics Centre, CEH Oxford, UK
24. Wu I-Ching, National Yang-Ming University, Taiwan

Appendix B

Participant Instructions for User Experiment 5

The following text includes verbatim the text that was distributed to the participants. The A1 posters made reference to in the text are included in reduced format in Appendices C and D.

myExperiment - Taverna workflow re-use exercise

Hello,

Many thanks for agreeing to participate in this exercise. Our goal is to find out how you re-use and repurpose workflows, selected from the myExperiment.org repository, to solve a set bioinformatics task. Your answers will greatly help us in building tools to support a "workflow by example" style to workflow authoring.

We will record how you go about the discovery and editing of workflows from the repository (referred to as "**candidate workflows**") in order to change a workflow you are thought to be working on ("**your workflow**"). All workflows were created with the Taverna workbench by twelve different authors.[1]

You are asked to solve tasks by making drawings and comments on paper diagrams of workflows. Most diagrams have been annotated with information about the task of a workflow and its services, both in natural language and with [semantic tags] from the ^{my}Grid bioinformatics service ontology (navigateable as a Web site at `http://www.mygrid.org.uk/ontology/OwlDoc/index.html`). This bundle also contains an appendix containing a list of common abbreviations.

Please do not be alarmed by the size of the exercise package :-) In a pilot study, the completion of the exercises took participants about two and a half hours. There are three steps:

1. (Important!) instructions on how to solve the exercises
2. A bundle of exercises, including a worked example
3. A survey about your experience

Please try to complete the exercises within two weeks of receiving them.

Many thanks again for your participation. We look forward to your answers and will keep you updated on your results and those of your colleagues!

Antoon Goderis, Franck Tanoh, Paul Fisher and Carole Goble
^{my}Grid/Taverna/ ^{my}Experiment team

1. Instructions on how to solve the exercises

The exercises are designed to capture a range of tasks which occur when re-using and integrating bioinformatics workflows. The intent is to have you create workflows that would be fully functional if you were to do this in a real workflow editor and not on paper.

There are two parts. In part 1, only the workflow diagram is shown. In part 2, documentation is provided. In each part, exercises are ordered by increasing difficulty. Afterwards, in a short survey, we will ask how long it took you in total to solve the entire batch of 11 exercises and poll for your experiences.

Now please take a look at the poster labelled "Example exercise."

[1] Our thanks to the Bioanalytical Sciences Group and Bio Health Informatics Group, U. Manchester; the Munich Information Center for Protein Sequences, Germany; National Cancer Research Institute, Italy; the Rice Group, European Bioinformatics Institute, UK and the SIGENAE team, INRA, France.

Each individual exercise is self contained on one poster.[2] It contains the workflow you are thought to be working on ("your workflow") and five candidate workflows. "Your workflow" is indicated by a circle around its number ("1" in the example poster). The poster also contains the task description at the bottom ("Based on workflow (1), obtain a list of gene identifiers. . . ") and a feedback box.

Each exercise has three steps to it:

A. The discovery step (indicated by "A" for each candidate workflow in the example),

B. The editing step ("B") and

C. The feedback step ("C").

A. The **discovery step**

This involves assigning a *relevance* assessment ("yes/no/maybe") to the candidate workflows (i.e. workflows 6, 8, 2, 12 and 17 in the example) with respect to how useful they are for solving the task given your workflow (i.e. workflow 1). In case you choose "maybe," please provide a reason. Note that multiple (or no) solutions may exist for an exercise, and that the combination of different candidate workflows may be needed to reach a solution.

B. The **editing step**

In this step you should *edit* a relevant workflow. This involves integrating a fragment of a relevant candidate workflow with your workflow.

Now please turn to the next poster in the bundle, labelled "Editing".

To solve a given task, you can edit workflows in different ways.

You will notice there are four possibilities to edit a workflow:

1. to put a workflow fragment at the end of your workflow,
2. to put it in front,
3. to replace a fragment or
4. to insert one.

Which of these cases is needed to solve a particular task will be up to you to decide!

During editing, you should do three things (as shown on the poster for each case):

1. Select the necessary and sufficient workflow fragment(-s) from a candidate workflow by circling it.
2. Connect it to your workflow by drawing the required connections between inputs

[2]Note that the exercise diagrams have been stripped from colour since colour reflects a Taverna Processor's type (Web service, local Java class,..), which does not matter for this exercise.

and outputs.

3. Cross out the parts that are no longer needed in your workflow.

To illustrate what the practical outcome of the editing action would be for each case, we have added the resulting workflow. Within the resulting workflow the bits coming from the candidate workflow are highlighted with a dotted box. As the exercise is performed on paper, you will only be making the connections and not see the resulting workflow on its own.

Now please return to the poster labelled "Example exercise."

You should be able to interpret the editing action between workflow 1 and workflow 2 now – it corresponds to the case of adding a fragment at the end of a workflow in the Editing poster.

C. The **feedback step**

After the editing step comes the feedback step. On the "Example exercise" poster, observe feedback step ("C"), which asks to assess the difficulty of the exercise and your confidence in solving it.

In addition, for the **second part** of the exercises you will be asked to specify how useful the available documentation and annotation was for doing the task:

1. "Indicate the workflows (if any) where the diagram *alone* provides enough information to determine whether the workflow is a solution *or not*."

List when and why you were able to determine from the diagram information alone whether a workflow was relevant and how it should be edited.

- E.g. none of the diagram boxes in workflow 12 had anything about BLAST which was enough for the exercise participant to decide against using it. She also decided that the information contained in the diagram of workflow 2 was enough to decide that that workflow was relevant and could be edited to solve the task (so in this case she decided in favour of using it).

2. "Indicate the workflows (if any) where the [semantic tagging] provides *essential* information to determine whether the workflow is a solution *or not*."

List when and why you found the semantic tagging to be indispensible for the outcome of the exercise, and it was not just replicating the natural language text.

- E.g. the participant felt that in service 3 on workflow 17, the semantic tag describing "Result" as a multiple sequence alignment report made it clear that "Result" was not a BLAST report and hence the workflow was irrelevant to solve the task.

Please feel free to write on the posters if you find places where, in your view, the documentation or annotation is confusing, wrong, or it is needed but missing.

That's it - you are now ready to start the exercises. Once you are finished, you can have the treat included in the tube ;-) If you are unclear about the instructions, please contact Antoon Goderis by phone on +44 161 275 0675 or e-mail on goderisa@cs.man.ac.uk. Good luck!

3. Exercises feedback and workflow re-use experiences
(please complete this after solving the exercises)

To complete the exercise, please go on-line and visit
http://myexperiment.org/exercise

4. Return the solutions

Please return the A1 sheets to us by post. Please make sure we can identify your package, by including your name or a business card somewhere.

Included in the tube you will find return postage as well as the return address on a sticky label. Please attach both to the tube.

Many thanks again for your participation. We will be in touch!

Antoon Goderis, Franck Tanoh, Paul Fisher and Carole Goble

List of common options/abbreviations

All fields (omitted from this appendix) with which the following databases can be queried: Ebi_medline2007, ebi_uniprot, ebi_srslinks

Shiftincrement

This is the amount by which the window is moved at each increment in order to find the melting point and other properties along the sequence.

windowsize

The values of melting point and other thermodynamic properties of the sequence are determined by taking a short length of sequence known as a window and determining the properties of the sequence in that window. The window is incrementally moved along the sequence with the properties being calculated at each new position.

Seqret parameter name: sformat . . . ..allowed values:

gcg, gcg8, embl, swiss, fasta, ncbi, genbank, nbrf, pir, codata, strider, clustal, phylip, acedb, msf, jackknifer, jackknifernon, nexus, nexusnon, treecon, mega, meganon, ig, staden, text, raw

Possible colour goviz web services : markTerm

(`http://www.graphviz.org/doc/info/colors.html`**)**

For the searchSimple service:

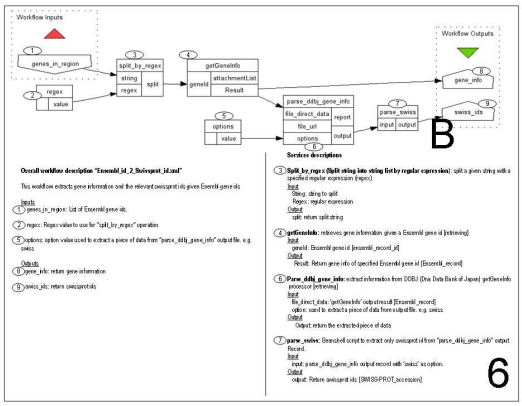
Parmeter name: database

Possible values: (omitted from this appendix)

Appendix C

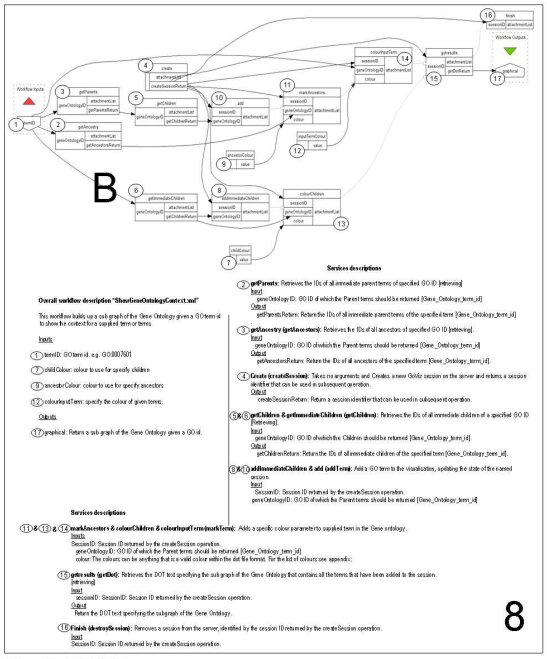
Example Workflow Exercise for User Experiment 5

Example exercise



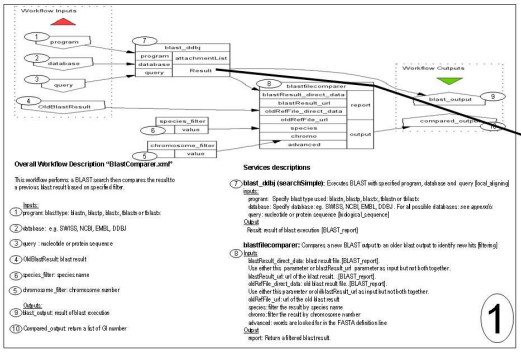
A

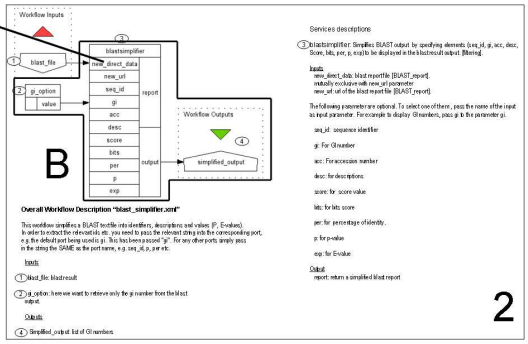
Relevant? Yes [ ] No [X] Maybe : _____



A

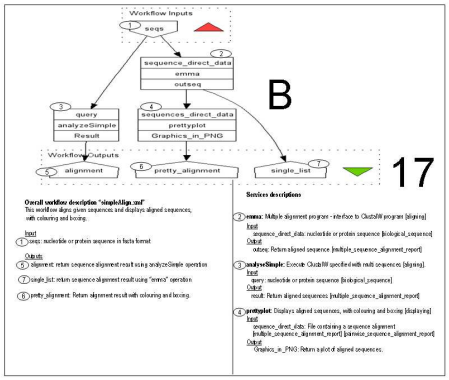
Relevant? Yes [ ] No [X] Maybe : _____





A

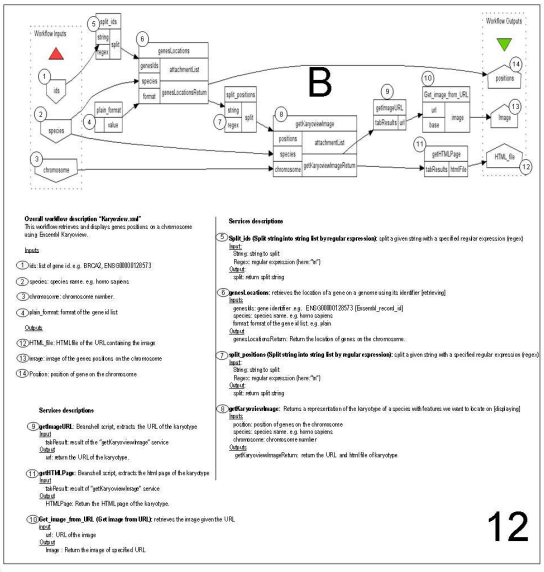
Relevant? Yes [ ] No [X] Maybe : _____



A

Relevant? Yes [ ] No [X] Maybe : _____

Based on workflow (1), obtain a list of gene identifiers by simplifying the BLAST output file.



A

Relevant? Yes [ ] No [X] Maybe : _____

C

Difficulty of task

Difficult [ ] Moderate [X] Easy [ ]

If difficult, please explain: _____

Confidence level in solution

High [X] Medium [ ] Low [ ]

If low, please explain: _____

Please indicate the workflows (if any) where the diagram **alone** provides enough information to determine whether it is a solution **or not**.

12: nothing involving BLAST 2: question is in workflow

Please indicate the workflows (if any) where the [semantic tagging] provides **essential** information to determine whether it is a solution **or not**.

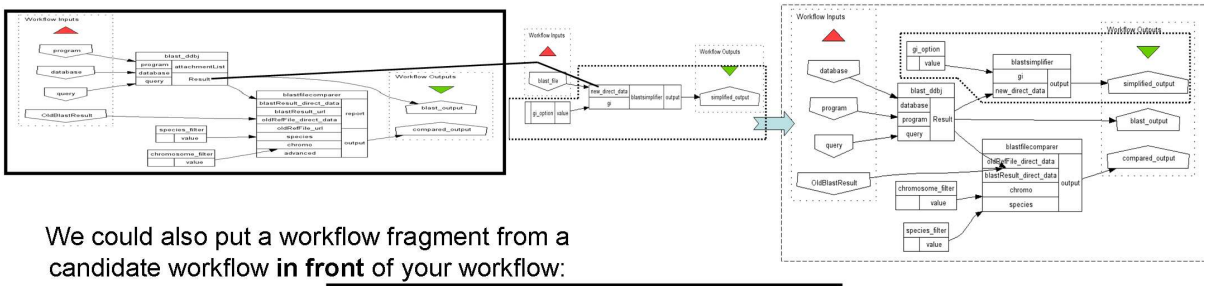
wf17:service3 "Result" is mult: seq. alignment report, not BLAST report

Appendix D

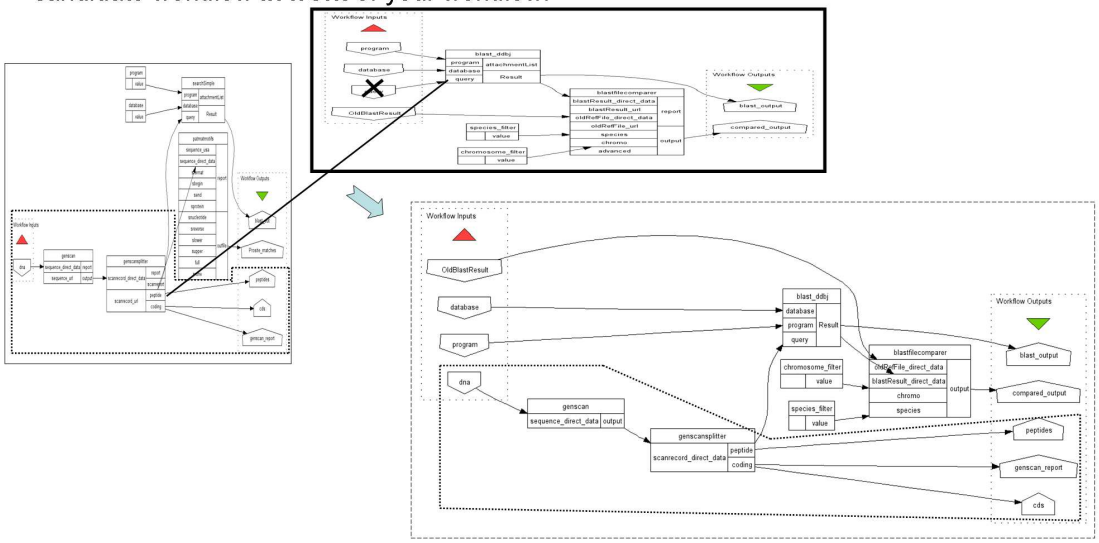
Possible Workflow Edit Operations for User Experiment 5

Editing

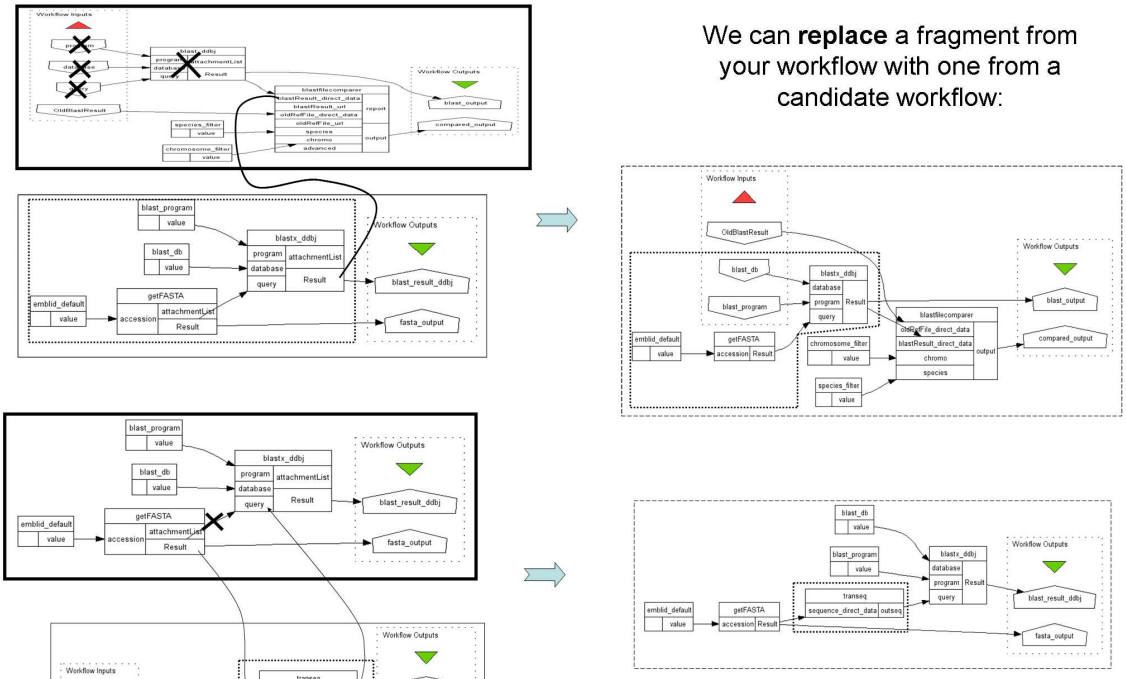
The following adds a fragment of a candidate workflow **at the end** of your workflow:



We could also put a workflow fragment from a candidate workflow **in front** of your workflow:



We can **replace** a fragment from your workflow with one from a candidate workflow:



To **insert** a fragment into your workflow, we connect the relevant inputs and outputs from a candidate workflow to the relevant outputs and inputs in yours.

Bibliography

[Agh86] G. Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge, MA, USA, 1986.

[BB91] A. Benveniste and G. Berry. The synchronous approach to reactive and real-time systems. *Proceedings of the IEEE*, 79:1270–1282, September 1991.

[BBS06] A. Basu, M. Bozga, and J. Sifakis. Modeling heterogeneous real-time components in bip. In *International Conference on Software Engineering and Formal Methods (SEFM)*, pages 3–12, Pune, 2006.

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel Schneider. *The Description Logic Handbook : Theory, Implementation and Applications*. Cambridge University Press, 2003.

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

[BEP06] Khalid Belhajjame, Suzanne M. Embury, and Norman W. Paton. On characterising and identifying mismatches in scientific workflows. In *DILS*, pages 240–247, 2006.

[BG98] Joe Bullock and Carole Goble. Tourist: the application of a description logic based semantic hypermedia system for tourism. In *9th ACM conference on Hypertext and hypermedia*, pages 132–141, 1998.

[BG99] S. Bechhofer and C. Goble. Classification Based Navigation and Retrieval for Picture Archives. In *IFIP WG2.6 Conference on Data Semantics, DS8*, Rotorua, New Zealand, January 1999.

[BGL+04] Daniela Berardi, Giuseppe De Giacomo, Maurizio Lenzerini, Massimo Mecella, and Diego Calvanese. Synthesis of underspecified composite e-services based on automated reasoning. In *2nd International Conference on Service Oriented Computing ICSOC*, pages 105–114. ACM Press, 2004.

[BHLM94] J. T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt. Ptolemy: A framework for simulating and prototyping heterogeneous systems. *Int. Journal of Computer Simulation, special issue on Simulation Software Development*, 4:155–182, 1994.

[BK02] Abraham Bernstein and Mark Klein. Towards high-precision service retrieval. In *Proceedings of the First International Semantic Web Conference (ISWC)*, Sardinia, Italy, 2002. Springer.

[BKBK05] A. Bernstein, E. Kaufmann, C. Brki, and M. Klein. How similar is it? towards personalized similarity measures in ontologies. In *7 Internationale Tagung Wirtschaftsinformatik*, February 2005.

[BKT02] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In *KR2002*, pages 203–214, San Francisco, USA, 2002.

[BLL+05] C. Brooks, E. A. Lee, X. Liu, S. Neuendorffer, Y. Zhao, and H. Zheng. Heterogeneous concurrent modeling and design in java. Tech. Report UCB/ERL M05/21, University of California, Berkeley, July 15 2005.

[BLNC06] S. Bowers, B. Ludaescher, A.H.H. Ngu, and T. Critchlow. Enabling scientific workflow reuse through structured composition of dataflow and control-flow. In *IEEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow)*, 2006.

[BMW03] M.-J. Blin, Claudia Bauzer Medeiros, and Jacques Wainer. A reuse-oriented workflow definition language. *Int. J. of Cooperative Information Systems*, 12(1):1–37, 2003.

[BP98] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *WWW7 / Computer Networks*, 30(1-7):107–117, 1998.

[BYRN99] Ricardo Baeza-Yates and Berthier Ribiero-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[CCC+04] A. Cali, D. Calvanese, S. Colucci, T. Di Noia, and F. M. Donini. A description logic based approach for matching user profiles. In *DL2004*, Whistler, British Columbia, Canada, 6-8 June 2004.

[CGB06] J. C. Corrales, D. Grigori, and M. Bouzeghoub. Bpel processes matchmaking for service discovery. In *Conference on Cooperative Information Systems (COOPIS)*, LNCS 4275, pages 237–254, Montpellier, France, 2006.

[CGL98] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *17th ACM SIGACT SIGMOD SIGART Symp PODS*, pages 149–158, 1998.

[CGWG07] V. Curcin, M. Ghanem, P. Wendel, and Y. Guo. Heterogeneous workflows in scientific workflow systems. In *Proc. of the 2nd Int. Workshop on Workflow Systems in e-Science (WSES 07) in conjunction with the Int. Conference on Computational Science (ICCS) 2007*, Beijing, China, May 27-30 2007.

[CNS+04] S. Colucci, T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. A uniform tableaux-based approach to concept abduction and contraction in aln. In *DL2004*, Whistler, British Columbia, Canada, 6-8 June 2004.

[DBG+04] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. H. Su, K. Vahi, and M. Livny. Pegasus: Mapping scientific workflow onto the grid. In *Across Grids Conference*, Nicosia, Cyprus, 2004.

[DHM+04] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity search for web services. In *Proc.s of the 30th VLDB Conference*, Toronto, Canada, 2004.

[DS04] M. Dean and G. Schreiber. OWL Web Ontology Language Reference. W3C Recommendation, World Wide Web Consortium. `www.w3.org/TR/owl-ref`, 2004.

[DS05] Schahram Dustdar and Wolfgang Schreiner. A survey on web services composition. *Int. J. Web and Grid Services*, 1(1), 2005.

[EJL+03] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity—the ptolemy approach. *Proceedings of the IEEE*, 91:127–144, January 2003.

[FHW+07] Paul Fisher, Cornelia Hedeler, Katherine Wolstencroft, Helen Hulme, Harry Noyes, Stephen Kemp, Robert Stevens, and Andrew Brass. A systematic strategy for large-scale analysis of genotype phenotype correlations: identification of candidate genes involved in african trypanosomiasis. *Nucleic Acids Research*, 35(16):5625–5633, August 2007.

[For04] Unicore Forum. Unicore plus final report: Uniform interface to computing resource. Technical report, `http://www.unicore.org/documents/UNICOREPlus-Final-Report.pdf`, 2004.

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

[GHS95] Dimitrios Georgakopoulos, Mark F. Hornick, and Amit P. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.

[GL05] C. Goble and B. Ludaescher, editors. *ACM Sigmod Record: Special Issue on Scientific Workflows*, volume 34. September 2005.

[GLL99] A. Girault, B. Lee, and E. A. Lee. Hierarchical finite state machines with multiple concurrency models. *IEEE Transactions On Computer-aided Design Of Integrated Circuits And Systems*, 18:742–760, 1999.

[Gol01] R. L. Goldstone. *MIT encyclopedia of the cognitive sciences*, chapter Similarity, pages 757–759. MIT Press, Cambridge, MA, 2001.

[GRD+07] Yolanda Gil, Varun Ratnakar, Ewa Deelman, Gaurang Mehta, and Jihie Kim. Wings for pegasus: Creating large-scale scientific applications using semantic representations of computational workflows. In *Proceedings of the 19th Annual Conference on Innovative Applications of Artificial Intelligence (IAAI)*, Vancouver, British Columbia, Canada, July 22-26 2007.

[Gro07a] Paul Groth. *The Origin of Data: Enabling the Determination of Provenance in Multi-institutional Scientific Systems through the Documentation of Processes*. Phd thesis, University of Southampton, 2007.

[Gro07b] GrADS Users Group. *GrADS Usersś Guide*. `http://www.iges.org/grads/gadoc/users.html`, Last accessed: December 2007 2007.

[Gro07c] GrADS Users Group. *GridFlow Online Manual*. `http://gridflow.ca/latest/doc/index.html`, Last accessed: December 2007 2007.

[GSLG05] Antoon Goderis, Ulrike Sattler, Phillip Lord, and Carole Goble. Seven bottlenecks to workflow reuse and repurposing. In *Int. Semantic Web Conference ISWC05*, volume 3792, pages 323–337, Galway, Ireland, 2005.

[GWS+02] R. Mark Greenwood, Chris Wroe, Robert Stevens, Carole A. Goble, and Matthew Addis. Position paper: Are bioinformaticians doing e-business? In *EuroWeb*, 2002.

[GZL07] M.-K. Leung G. Zhou and E. A. Lee. A code generation framework for actor-oriented models with partial evaluation. In *International Conference on Embedded Software and Systems (ICESS)*, LNCS 4523, pages 786–799, Daegu, Korea, 2007. Springer.

[Har87] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[HB07] C. Hardebolle and F. Boulanger. Modhel'x: A component-oriented approach to multi- formalism modeling. In *MODELS 2007 workshop on Multi- Paradigm Modeling*, Nashville, Tennessee, USA, 2007.

[Hew77] C. Hewitt. Viewing control structures as patterns of passing messages. *Journal of Artificial Intelligence*, 8:323–363, 1977.

[HPSvH03] I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[HS03] I. Horrocks and U. Sattler. Decidability of SHIQ with Complex Role Inclusion Axioms. In *IJCAI-03*, Acapulco, Mexico, 2003.

[HT02] Tony Hey and Anne Trefethen. The uk e-science core program and the grid. In *International Conference on Computational Science*, volume 1, pages 3–21, 2002.

[HZB+06] D. Hull, E. Zolin, A. Bovykin, I. Horrocks, U. Sattler, and R. Stevens. Deciding semantic matching of stateless services. In *Proc. of AAAI'2006*, 2006.

[JH98] Gregor Joeris and Otthein Herzog. Managing evolving workflow specifications. In *3rd IFCIS International Conference on Cooperative Information Systems (CoopIS98)*, pages 310–319, New York, August 1998.

[JS05] A. Jantsch and I. Sander. Models of computation and languages for embedded system design. *IEEE Proceedings on Computers and Digital Techniques*, 152:114–129, March 2005.

[Kar03] Nikiforos Karamanis. *Entity Coherence for Descriptive Text Structuring*. Phd thesis, School of Informatics, University of Edinburgh, 2003.

[KBL+07] Christoph Kiefer, Abraham Bernstein, Hong Joo Lee, Mark Klein, and Markus Stocker. Semantic process retrieval with iSPARQL. In *European Semantic Web Conference (ESWC)*, pages 609–623, 2007.

[KGR06] J. Kim, Y. Gil, and V. Ratnakar. Semantic metadata generation for large scientific workflows. In *Int. Semantic Web Conference (ISWC)*, Athens, USA, November 5-9 2006.

[KLP] D. Koller, A. Levy, and A. Pfeffer. P-classic: A tractable probabilistic description logic. In *AAAI 1997*, pages 390–397, Rhode Island, August.

[KLP+04] U. Keller, R. Lara, A. Polleres, et al. Wsmo web service discovery. WSML Working Draft D5.1 v0.1, University of Innsbruck, 12 November 2004.

[KM77] G. Kahn and D. B. MacQueen. *Information Processing*, chapter Coroutines and Networks of Parallel Processes. North-Holland Publishing Co, 1977.

[Kru92] Charles W. Krueger. Software reuse. *ACM Comput. Surv.*, 24(2), 1992.

[KWJ+04] R. King, K. Whelan, F. Jones, P. Reiser, C. Bryant, S. Muggleton, D. Kell, and S. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(247-252), 2004.

[LAB+05] B. Ludaescher, I. Altintas, C. Berkley, et al. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience, Special Issue on Scientific Workflows*, 2005.

[Las05] G. Von Laszewski. Java cog kit workflow concepts for scientific experiments. Technical report, Argonne National Laboratory, Argonne, IL, USA, 2005.

[LAWG05] Phillip Lord, Pinar Alper, Chris Wroe, and Carole Goble. Feta: A light-weight architecture for user oriented semantic service discovery. In *European Semantic Web Conference*, 2005. Accepted for Publication.

[LBW+04] Phillip Lord, Sean Bechhofer, Mark D. Wilkinson, Gary Schiltz, Damian Gessler, Duncan Hull, Carole Goble, and Lincoln Stein. Applying semantic web services to bioinformatics: Experiences gained, lessons learnt. In *International Semantic Web Conference*, pages 350–364, 2004.

[LHJ+04] P. Li, K. Hayward, C. Jennings, K. Owen, T. Oinn, R. Stevens, S. Pearce, and A. Wipat. Association of variations in i kappa b-epsilon with graves' disease using classical and mygrid methodologies. In *Proc UK e-Science All Hands Meeting*, Nottingham, 31st August - 3rd September 2004.

[LPA06] Chen Y. Li P. and Romanovsky A. Measuring the dependability of web services for use in e-science experiments. In *3rd International Service Availability Symposium*, Helsinki, Finland, May 15-16 2006.

[LR05] David Lambert and David Robertson. Matchmaking multi-party interactions using historical performance data. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 611–617, New York, NY, USA, 2005. ACM.

[LSV98] E. A. Lee and A. Sangiovanni-Vincentelli. A framework for comparing models of computation. *IEEE Transactions on CAD*, 17(12), December 1998.

[LZ07] E. A. Lee and H. Zheng. Leveraging synchronous language principles for heterogeneous modeling and design of embedded systems. In *EMSOFT*, Salzburg, Austria, October 2007.

[MB00] B.T. Messmer and H. Bunke. Efficient subgraph isomorphism detection: a decomposition approach. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):307–323, Mar/Apr 2000.

[MB07] F. Maraninchi and T. Bhouhadiba. 42: Programmable models of computation for a component-based approach to heterogeneous embedded systems. In *6th ACM International Conference on Generative Programming and Component Engineering (GPCE)*, Salzburg, Austria, 2007.

[MBE03] Brahim Medjahed, Athman Bouguettaya, and Ahmed K. Elmagarmid. Composing web services on the semantic web. *VLDB J.*, 12(4), 2003.

[MCH03] Thomas W. Malone, Kevin Crowston, and George A. Herman, editors. *Organizing Business Knowledge: The MIT Process Handbook*. MIT Press, 2003.

[MHH] D. Miers, P. Harmon, and C. Hall. The 2007 bpm suites report. http://www.bptrends.com.

[MM03] Sheila A. McIlraith and David L. Martin. Bringing semantics to web services. *IEEE Intelligent Systems*, January - February 2003.

[MPAD+05] C. B. Medeiros, J. Perez-Alcazar, L. Digiampietri, G. Z. Pastorello Jr, A. Santanche, R. S. Torres, E. Madeira, and E. Bacarin. Woodss and the web: Annotating and reusing scientific workflows. *SIGMOD Record Special Issue on Scientific Workflows*, 34(3), September 2005.

[MSST93] C. Meghini, F. Sebastiani, U. Straccia, and C. Thanos. A model of information retrieval based on a terminological logic. In *116th ACM SIGIR*, pages 298 – 307, Pittsburgh, Pennsylvania, 1993.

[MW06] Bendick Mahleko and Andreas Wombacher. Indexing business processes based on annotated finite state automata. In *ICWS*, pages 303–311, 2006.

[MWG04] Shalil Majithia, David W. Walker, and W.A. Gray. Automated web service composition using semantic web technologies. In *Proceedings of the International Conference on Autonomic Computing (ICAC'04)*, New York, New York, 17 - 18 May 2004.

[MYA+04] Stephen McGough, Laurie Young, Ali Afzal, Steven Newhouse, and John Darlington. Workflow enactment in iceni. In *Proceedings of Fourth All Hands Meeting (AHM04)*, Nottingham, 20 - 22nd September 2004.

[NHG06] Falk Neubauer, Andreas Hoheisel, and Joachim Geiler. Workflow-based grid applications. *Future Generation Computer Systems*, 22(1-2):6–15, 2006.

[OGA+05] Tom Oinn, Mark Greenwood, Matthew Addis, Nedim Alpdemir, Justin Ferris, Kevin Glover, Carole Goble, Antoon Goderis, Duncan Hull, Darren Marvin, Peter Li, Phillip Lord, Matthew Pocock, Martin Senger, Robert Stevens, Anil Wipat, and Chris Wroe. Taverna: Lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience: Special Issue on Scientific Workflows*, 2005.

[Oin07] T. Oinn. Taverna 2 workflow specification. `www.ebi.ac.uk/~tmo/docs/t2semantics.pdf`, Last accessed 28 September 2007.

[PA04] Cesare Pautasso and Gustavo Alonso. Jopera: a toolkit for efficient visual composition of web services. *International Journal of Electronic Commerce (IJEC)*, 9(2), 2004.

[PGM97] John Park, John Gennari, and Mark Musen. Mappings for reuse in knowledge-based systems. Technical Report 97-0697, Stanford Medical Informatics, 1997.

[PS04] H. D. Patel and S. K. Shukla. *SystemC Kernel Extensions for Heterogeneous System Modelling*. Kluwer, 2004.

[RS04] Jinghai Rao and Xiaomeng Su. A survey of automated web service composition methods. In *SWSWPC*, pages 43–54, 2004.

[RtHvdAM06] N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar. Workflow control-flow patterns: A revised view. BPM Center Report BPM-06-22, 2006.

[SAB06] W. Sudholt, I. Altintas, and K.K. Baldridge. A scientific workflow infrastructure for computational chemistry on the grid. In *1st Int. Workshop on Computational Chemistry and Its Application in e-Science in conjunction with ICCS*, 2006.

[SC88] S. Siegel and J. N. Castellan. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, 1988.

[SEG+03] S. Al Sairaf, F. S. Emmanouil, M. Ghanem, N. Giannadakis, Y. Guo, D. Kalaitzopolous, M. Osmond, A. Rowe, iJ. Syed, and P. Wendel. The design of discovery net: Towards open grid services for knowledge discovery. *International Journal of High Performance Computing Applications*, 2003.

[SPAS03] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. Automated discovery, interaction and composition of semantic web services. *Web Semantics: Science, Services and Agents on the WWW*, 1(1):27–46, 2003.

[STW+04] R.D. Stevens, H.J. Tipney, C.J. Wroe, T.M. Oinn, M. Senger, P.W. Lord, C.A. Goble, A. Brass, and M. Tassabehji. Exploring Williams Beuren Syndrome Using myGrid. *Bioinformatics*, 20:303–310, 2004.

[SVK+07] Carlos E. Scheidegger, Huy T. Vo, David Koop, Juliana Freire, and Claudio T. Silva. Querying and creating visualizations by analogy. *IEEE Trans. Vis. Comp. Graph.*, 13(6):1560–1567, 2007.

[SVW06] Rajkumar Buyya Srikumar Venugopal and Lyle Winton. A grid service broker for scheduling e-science applications on global data grids. *Concurrency and Computation: Practice and Experience*, 18(6):685–699, May 2006.

[TCS+04] F. Tao, L. Chen, N. Shadbolt, et al. Semantic web based content enrichment and knowledge reuse in e-science. In *CoopIS/DOA/ODBASE*, pages 654–669, 2004.

[TIR+07] Ioan Toma, Kashif Iqbal, Dumitru Roman, Thomas Strang, Dieter Fensel, Brahmananda Sapkota, Matthew Moran, and Juan Miguel Gomez. Discovery in grid and web services environments: A survey and evaluation. *Multiagent and Grid Systems Special Issue on Advances in Grid services Engineering and Management*, 3(3):341–352, 2007.

[Tur06] D. Turi. Taverna workflows: Syntax and semantics. `http://www.mygrid.org.uk/wiki/Mygrid/TavernaSemantics`, May 23 2006.

[Tve77] A. Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.

[TWML01] Todd Tannenbaum, Derek Wright, Karen Miller, and Miron Livny. Condor – a distributed job scheduler. In Thomas Sterling, editor, *Beowulf Cluster Computing with Linux*. MIT Press, October 2001.

[vdA05] W.M.P. van der Aalst. Process mining in cscw systems. In *Ninth Int. Conference on Computer Supported Cooperative Work in Design*, May 2005.

[vdAtHKB03] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[WEB+07] B. Wassermann, W. Emmerich, B. Butchart, N. Cameron, L. Chen, and J. Patel. *Workflows for e-Science*, chapter Sedna: A BPEL-based environment for visual scientific workflow modelling, pages 428–449. Springer, 2007.

[Wen06] Keith Wenson. The bpmn-xpdl-bpel value chain. Blog entry; `http://kswenson.wordpress.com/2006/05/26/bpmn-xpdl-and-bpel`, May 26 2006.

[WGG+04] Chris Wroe, Carole Goble, Mark Greenwood, Phillip Lord, Simon Miles, Juri Papay, Terry Payne, and Luc Moreau. Automating experiments using semantic data on a bioinformatics grid. *IEEE Intelligent Systems*, January-February 2004.

[WGG+07] Chris Wroe, Carole Goble, Antoon Goderis, Phillip Lord, Simon Miles, Juri Papay, Pinar Alper, and Luc Moreau. Recycling workflows and services through discovery and reuse: Research articles. *Concurr. Comput. : Pract. Exper.*, 19(2):181–194, 2007.

[Wom06] A. Wombacher. Evaluation of technical measures for workflow similarity based on a pilot study. In *CoopIS*, Montpellier, France, November 1-3 2006.

[WPF05] Marek Wieczorek, Radu Prodan, and Thomas Fahringer. Scheduling of Scientific Workflows in the ASKALON Grid Environment. *ACM SIGMOD Record*, 35(3), 2005. http://dps.uibk.ac.at/ marek/publications/acm-sigmod-2005.pdf.

[WR06] A. Wombacher and M. Rozie. Piloting an empirical study on measures for workflow similarity. In *IEEE Int. Conference on Service Computing (SCC)*, Chicago, USA, September 18-22 2006.

[WSG+03] C. Wroe, R. Stevens, C. Goble, A. Roberts, and M. Greenwood. A suite of daml+oil ontologies to describe bioinformatics web services and data. *Intl. J. of Cooperative Information Systems*, 12(2):197–224, 2003.

[YB05] Jia Yu and Rajkumar Buyya. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing*, 3(3):171–200, 9 2005.

[ZBB+06] Zhiming Zhao, Suresh Booms, Adam Belloum, Cees de Laat, and Bob Hertzberger. Vle-wfbus: A scientific workflow bus for multi e-science domains. In *E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, page 11, Washington, DC, USA, 2006. IEEE Computer Society.

[Zha07] Jun Zhao. *A conceptual model for e-science provenance*. Phd thesis, University of Manchester, 2007.

[ZWF06] Y. Zhao, M. Wilde, and I. Foster. Applying the virtual data provenance model. In *Int. Provenance and Annotation Workshop (IPAW)*, Chicago, USA, May 3-5 2006.

[ZWG+04] Jun Zhao, Chris Wroe, Carole Goble, Robert Stevens, Dennis Quan, and Mark Greenwood. Using semantic web technologies for representing e-science provenance. In *Third International Semantic Web Conference*, Hiroshima, Japan, 2004.