Seven Bottlenecks to Workflow Reuse and Repurposing

Antoon Goderis, Ulrike Sattler, Phillip Lord, and Carole Goble

School of Computer Science, University of Manchester, UK
{goderis, carole, sattler, plord}@cs.man.ac.uk

Abstract. To date on-line processes (*i.e.* workflows) built in e-Science have been the result of collaborative team efforts. As more of these workflows are built, scientists start sharing and reusing stand-alone compositions of services, or *workflow fragments*. They *repurpose* an existing 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 it. Such a "workflow by example" approach complements the popular view in the Semantic Web Services literature that on-line processes are constructed automatically from scratch, and could help bootstrap the Web of Science. Based on a comparison of e-Science middleware projects, this paper identifies seven bottlenecks to scalable reuse and repurposing. We include some thoughts on the applicability of using OWL for two bottlenecks: workflow fragment discovery and the ranking of fragments.

1 Towards a Web of Science

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 Web and Grid services, 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 middleware layer for scientists, thereby aiming to increase collaboration within and between scientific fields [1]. 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 (www.mygrid.org.uk) workbench, a set of components to build workflows in bioinformatics, currently allows access to over thirteen hundred distributed services and has produced over a hundred workflows, some of which orchestrate up to fifty services. These resources have been developed by users and service providers distributed throughout the global biology community. Figure 1 shows an example of a ^{my}Grid workflow which gathers information about genetic sequences in support of research on Williams Beuren syndrome (WBS) [2].

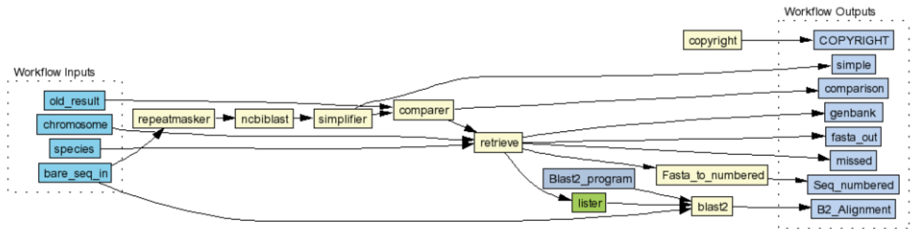

Fig. 1. Part of a ^{my}Grid workflow to annotate genetic sequences as presented by the ^{my}Grid Taverna workbench. The diagram shows the typical fanning out behaviour of a bioinformatics pipeline, producing lots of data from a limited number of inputs (the left and right boxes) based on a set of distributed services (the middle boxes).

We are now witnessing how scientists have started reusing and propagating *in silico* experiments as commodities and "know-how" in their own right. To cater for the reuse of *in silico* experiments on the scale of the Web of Science [3], the e-Science infrastructure will need to expand its current handling of the workflow life cycle. The goal of this paper is to investigate how reuse and repurposing of *in silico* experiments would work. We see reuse and repurposing as a way of bootstrapping the Web of Science by stimulating the dynamics of sharing and reusing experimental components in the scientific community. Section 2 highlights the benefits of workflow reuse, distinguishes between workflow reuse and repurposing, and analyses the relationship with related work. Section 3 goes bottom up, showcasing different types of reuse based on case studies from e-Science middleware projects. From this survey, we obtain the following seven bottlenecks to reuse and repurposing, which are presented in Section 4. For bottlenecks 5 and 7, we consider how reasoning over ontologies in the Web Ontology Language (OWL) [4] could widen them.

1. Restrictions on service availability
2. Rigidity of service and workflow language definitions
3. Intellectual property rights on workflows
4. Workflow interoperability
5. Lack of a comprehensive discovery model
6. The process knowledge acquisition bottleneck
7. Lack of workflow fragment rankings

The bottlenecks belong to two broad categories. First, some bottlenecks hinder establishing a critical mass for bootstrapping the Web of Science. Bottlenecks 1-4 identify reasons why we do not have as many workflows available for reuse as we might expect. Provided that this set of bottlenecks can be suitable addressed, the available pool of workflows then still needs to be easily searchable and adaptable. Bottlenecks 5-7 identify barriers that keep people from effectively processing the available workflow knowledge.

Bottlenecks 4-7 are closely related to challenges for the Semantic Web, cited in [5] and marked up in italics below. In particular, to maximise the available base of workflows, one would need to resolve workflow language interoperability issues. Workflow interoperability in essence seeks to interoperate different conceptualisations of control flow, a special case of *reconciling different conceptualisations* of a domain. The *tradeoff in Knowledge Representation* between expressivity and tractability relates to the current lack of a comprehensive model for discovering fragments. The building and populating of a comprehensive model is subject to the process *knowledge acquisition bottleneck*. Finally, to fully exploit the resulting model and its contents, the reuse infrastructure should support *unpredictable use of knowledge*, *e.g.* through rankings for fragments dependent on a user's context.

2 Reuse and Repurposing in e-Science

e-Scientists are driven by a desire to set up and run *in silico* experiments which complement the work done in the laboratory. As more workflows are built, scientists start sharing and reusing stand-alone compositions of services, or *workflow fragments*, within and between research projects. As a result, scientists are adopting a "workflow by example" style of workflow construction by reusing and repurposing existing experience. This complements the vision that experiments could be composed automatically, *e.g.* the Robot Scientist [6].

2.1 Why Workflow Reuse?

Workflow reuse in e-Science is intrinsically linked to a desire that workflows be shared and reused by the community as best practice scientific protocols or know-how. It has the potential to: reduce workflow authoring time (less re-inventing the wheel); improve quality through shared workflow development (two heads are better than one, or leveraging the expertise of previous users); and improve experimental provenance at the process level through reuse of established and validated workflows (analogous to using proven algorithms or practices rather than inventing a new, and potentially error-prone, one yourself). Concretely, the research group who produced the Williams' syndrome workflow [2] have already seen a dramatic drop in workflow authoring time through the ability to repurpose *workflow fragments* from previous experiments.

A *workflow fragment* is a piece of an experimental description that is a coherent sub-workflow that makes sense to a domain specialist. It is a snippet of workflow code written in a workflow orchestration language which typically carries annotation to facilitate its discovery. Each fragment forms a useful resource in its own right and is identified at publication time.

2.2 Reuse and Repurposing

We distinguish between *reuse*, where workflows and workflow fragments created by one user might be used as is, and *repurposing*, where they are used as a starting point by others.

- A user will reuse 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.

Repurposing requires 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." The techniques work off a knowledge base of existing workflows (either a central registry or a peer to peer setting). The end result, a repurposed workflow, is contributed back to the pool of available know-how.

Of the workflows produced by the projects surveyed in Section 3, many model simple pipelines like the one in Figure 1 but some also model complex concurrent control flows. Based on frequent interaction with domain scientists, we adopt the *working hypothesis* that a *scientist* thinks about her workflow primarily as data flow, transforming scientific data sets, and ignores what might be going on under the hood in terms of complex control flow. As a result, a scientist is interested in making queries that involve discovery of fragments based on data, services, and at most involve simple ordering, choice points and loops. We have collected a set of practical reuse and repurposing queries Q1-Q7 for domain scientists. The use of semantics seems relevant to solve queries Q1-Q5; we revisit them in Sections 4.5 and 4.7. For reasons of scope we leave aside Q6 and Q7 in this paper.

Q1 Given a data point, service, fragment or workflow, where has this item been used before?
Q2 Show the common data, services, and compositions of services and data between two workflows or fragments.
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 workflows become available, fragments are reused and repurposed in a variety of workflows. How can one systematically keep track of these interrelationships?
Q5 Since the design and implementation of a workflow can extend over long periods of time (months, even years), one might want to store even partially described workflows. Which are the available workflows in progress?
Q6 Show the differences between two workflow versions.
Q7 Show the evolution of a workflow over time.

Conversely, an advanced *workflow developer* typically implements complex distributed processes involving concurrency and has little affinity with scientific jargon. Developers might also ask queries like the above, but these would not involve jargon. In those cases where developers build *complex control flows*, they typically work by example, and as such might issue queries for examples of implemented complex flows. Typically there would be interaction between the two user roles during workflow construction, as part of a collaborative effort.

2.3 Repurposing, Discovery and Composition

How does repurposing relate to service discovery and composition? We answer this by first outlining the different aspects of the service life cycle and then characterising repurposing in these terms. Web-enabled services, whether published as Web, Grid or peer to peer services 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 [7]. Based on such descriptions, services can be *discovered, composed, configured, verified, simulated, invoked* and *monitored.* Of these, discovery and composition are the most relevant for repurposing.

- Discovery is the process of finding, ranking and selecting *existing* services. Discovery can be exact or inexact, and operates over descriptions of atomic or composite services which consist of atomic services.
- Composition is the process of combining services into a *new* working assembly. It is performed either manually, semi-automatically or automatically. Composition typically combines service discovery with service integration. If either activity involves manual intervention from a human, composition becomes non-automated.

Mapping repurposing to this classification yields the following distinctive set of features:

Workflow fragments, not services on the Web. Workflow fragments orchestrate services located on the Web. Fragments are not Web-enabled services, however, in the sense that they can be readily invoked over the Web. Instead they require a workflow engine for execution. At an abstract level, fragments can be regarded as composite services, which means some of the formal language machinery being developed for Web-enabled services is still applicable.

Behavioural service descriptions. Fragments are snippets of code published in a workflow language which typically carry annotation to facilitate their discovery. Fragments can describe sophisticated forms of control flow between services.

Design level discovery over composite services. In general, the literature on discovering composite services/ processes is investigating discovery at three levels. For each level, an example of queries is shown for which techniques are available.

Design level discovery. Scientists may ask questions that comprise simple structural elements, such as relating to parts of a process, *e.g.* [8], or loops and choice points *e.g.* [9], whereas developers may pose queries relating to complex control flow, such as dealing with constraints on messaging behaviour *e.g.* [10] or distributed execution models [11].

Enactment level discovery. For instance, based on feedback on the behaviour of particular components during the run of a process, a user may select a new, similar process that is more likely to achieve the stated goal [12].

Post-enactment level discovery. Process languages sometimes allow for great flexibility in the execution path a user can choose. Several authors consider process mining, which seeks to discover from the enactment data of workflow runs, which path users actually follow in practice *e.g.* [13].

With respect to fragment discovery, we are only concerned with design level discovery in order to retrieve snippets of workflow code.

Exact and inexact discovery. Lacking a sufficient set of answers based on exact discovery, repurposing techniques can progress to inexact discovery techniques, which find the closest available alternatives (for a human to then look at). Sections 4.5 and 4.7 discuss some available options based on OWL. A distinctive feature of repurposing techniques is the inclusion of a measure of *integration effort* in the rankings of returned fragments.

Semi-automatic composition. Composition combines service discovery and service integration. Repurposing a workflow based on workflow fragments relies on automated support for the *discovery* part, which generates clues as to what would be the best fragments for a human to consider based on the *existing* workflows. The actual integration part is left up to the workflow developer. Hence a newly repurposed workflow is the result of semi-automatic *composition.* Repurposing techniques in this sense are to be seen as *composition-oriented discovery* techniques and sit in between automated discovery and automated composition. We draw on the observation made in [14] that scientists in general are reluctant to relinquish control over the construction of their experiments. We aim to support scientists' activities, not replace them.

2.4 Abstract and Concrete Workflows

Various authors in the scientific workflow literature use the notion of abstract and concrete workflows [15]. The notion is useful for repurposing as it helps to create a view over aspects of a workflow that either a domain scientist or a developer are interested in. *Abstract* workflows capture a layer of process description that abstracts away from the task and behaviour of a *concrete* workflow. The kinds of abstraction performed are a modelling decision and depend on the application. Generally speaking, abstractions can generalise over:

1. Workflow and service parameters (task, parameters, data, component services): these abstract workflows have also been called *workflow templates* [15]. Templates are un-invocable, un-parameterised workflows whose services are unbound to a specific end point.
2. Control constructs: such abstract workflows can be organised based on *workflow patterns* [16] and distributed execution models [11].
3. Domain specificity: abstract workflows like these focus on capturing problem-solving behaviour and are the subject of *Problem-Solving Methods* research [12].

The distinction between abstract and concrete workflows is useful for at least three types of applications. Firstly, abstract workflows can guide the *configuration* of generic pieces of workflow into concrete workflows. The end result is a concrete workflow like the one depicted on Figure 1. Secondly, the notion of abstract workflows is useful for *dynamic bindings for scheduling and planning*, where service availability changes frequently and one queries for run-time instantiations of service classes. Thirdly, one can use the abstract-concrete distinction to support queries for *repurposing*. In particular, the first type of abstraction layer, over workflow parameters can be used to support queries Q1-Q5 (see [17] for details). The second type of abstraction layer, over control constructs, serves to answer developer's queries for complex control flow.

3 Scientific Workflow Reuse in Practice

After defining reuse and repurposing and contrasting it with related work, we now take a bottom up approach. We ask the question how far off we are at the current time from a Web of Science enabled by reuse and repurposing. As more scientists start to construct workflows, opportunities for cross-fertilisation are likely to arise. We present the results of a survey on how reuse and repurposing occurs in practice.

3.1 Case Studies in Workflow Fragment Reuse

We take a cross-section of middleware projects from the e-Science programme in the United Kingdom, which was the first of its kind [1]. To collect case studies of reuse, we have collaborated with biologists and developers in the ^{my}Grid project and interviewed core developers from the UK-based InforSense (the commercial collaborator of the DiscoveryNet e-Science project [18]), Geodise [19], Triana [20] and Sedna projects. We also interviewed people from the USA-based Kepler project [21]. The following case studies arose from the interviews.[1]

- In ^{my}Grid, around 200 users have built 100 workflows from over 1300 services. Workflow fragments have been repurposed between different research groups in Manchester, Newcastle and Liverpool investigating Williams' syndrome [2], Graves' disease [22] and Trypanosomiasis (sleeping sickness) in cattle, respectively (see Figure 2). New reuse of fragments from the Williams workflow is planned to support research on the Aspergillus fungus.
- 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.
- 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 very limited.

[1] The survey form is available from `www.cs.man.ac.uk/~goderisa/surveyform.pdf`

- 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.
- 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 reuses both configurations and assemblies of Matlab functions (*i.e.* scripts) described by various authors.
- The Sedna project at University College London has built a compute intensive workflow for chemistry, generating up to 1200 service instances concurrently. No reuse of this workflow has occurred. Sedna is notable as it is the only project in our sample to use BPEL (Business Process Execution Language), which is considered a *de facto* standard for business workflows.

3.2 Three Kinds of Workflow Reuse

From these use cases, three categories of workflow reuse surfaced, which are based on the person doing the reuse: reuse by third parties who the workflow author never met, reuse by collaborators, and personal reuse.

Reuse by third parties. Third-party reuse is the kind of reuse envisaged by the e-Science vision for inter-disciplinary scientific collaboration. None of the interviewees could report reuse of this kind.

Reuse by collaborators. Scientists are typically part of a research group and various research projects, inside of which they exchange knowledge. Figure 2 shows the reuse of fragments between research groups active in the same project (from Graves to Williams), as well as reuse 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 reuse 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.*

Personal reuse. 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 reuse". In the case of the Graves workflow, the workflow took more than a year to create. During the process of building it, 56 bits of workflow were created, most of which are overlapping versions and used in one shape or other in the other versions. The largest of these bits contains 45 elements, not counting the links between the data and services.

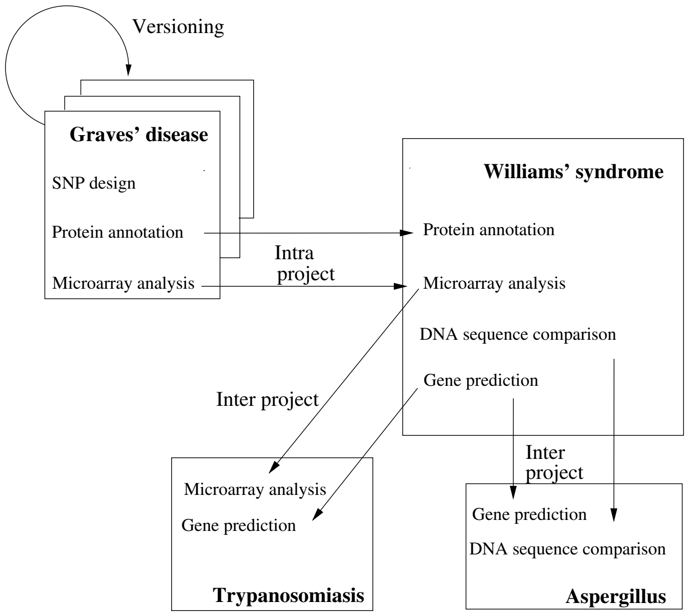

Fig. 2. Different types of workflow reuse illustrated by a scenario from bioinformatics

The picture for the bioinformatics use case in Figure 2 does not do justice to the difficulty it took to reuse 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 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.

One can conclude that reuse and *manual* repurposing of workflows and fragments of workflows is already happening. It is clear however that reuse becomes harder as the conceptual and physical distance between parties increases. If reuse and repurposing 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.

4 Seven Bottlenecks to Reuse and Repurposing

Based on the comparison of e-Science middleware projects, we can identify seven bottlenecks to scalable reuse and repurposing. The bottlenecks belong to two broad groups: those preventing the collection of a large pool of workflows, and those that prevent discovery of workflow fragments in that available pool of knowledge. Identifying and addressing the first group is critical to establishing a Web of Science: without a substantial pool of workflows, there cannot be a Web of Science as there will be no scientific components to annotate and query for.

4.1 Restrictions on Service Availability

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 favourite 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 except Sedna offer access to types of services which are other than plain 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.

4.2 Rigidity of Service and Workflow Language Definitions

Services on the Web typically are outside the control of a workflow developer. The presented *service interface defines the limit* to which one can reuse 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 reuse in mind by providing rich interfaces.

Workflow specifications can be hard to reuse too, 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 [23]. 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.

4.3 Intellectual Property Rights on Workflows

Scientists invest a lot of time in building workflows and are often *hesitant to release workflows* without formal Intellectual Property Rights agreements. We have seen this with the Williams, Graves and InforSense workflows. 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.

4.4 Workflow Interoperability

The saying "The nice thing about standards is that you can choose" also holds for scientific workflow languages. Each of the projects in the survey uses its own language for orchestrating resources. This diversity reflects the different demands of the application areas and computer skills of users. For repurposing, it is desirable to have access to as wide a pool of workflows as possible. Libraries of workflow *patterns* for control flow [16] have been developed and used to compare commercial workflow software for business processes. To our knowledge, this work has not been applied to compare and inform interoperation between scientific workflow systems. Also, these patterns do not address how combinations of patterns result into distributed execution models. In particular, developers would want to know how such models *compare* and can be *combined.* Kepler for instance have built workflows for environmental modelling that combines different distributed execution models (called "Directors" [11]) in one specification.

4.5 Lack of a Comprehensive Discovery Model

Designing representations for *in silico* experiments that can capture what is being done, why, and what has been tried before but failed, is a big challenge. Here we focus on the kind of information needed in such a representation to support discovery of fragments.

We have noted in Section 2.4 that different abstraction layers can be used to discover workflow fragments. Our *hypothesis* is that workflow fragment discovery requires the use of control flow constructs. How rich the control flow query support should be depends on the envisaged user (as explained in Section 2.2).

Unfortunately no one formalism can be expected to support all the desired control flow queries. We reflect on whether the Web Ontology Language OWL [4] could be used for searching workflow fragments. We consider this expressive Description Logic (DL) because of: (i) it being a standardised KR language; (ii) the support it offers for classifying a large collection of instances, *e.g.* workflow fragments; (iii) the potential to describe and query for workflows at a level of abstraction suited for a domain scientist through query languages; (iv) the support for representing incomplete workflows. OWL should be well suited to formulate data flow queries pertaining to inputs and outputs of services. DL ontologies in general are limited to modelling simple control flow constructs, however. Other formalisms provide a better fit for querying for complex control flow (*e.g.* process algebras). Though ideally one would like to be able to combine complex data flow queries with complex control flow queries, given the complexity of the task, we will first try to combine the outcome of querying different formalisms and present this in a uniform manner to the user.

Could we use OWL annotation to answer the data flow queries Q1-Q5 of Section 2.2? Various authors have experimented before with *service* discovery using DL reasoning, typically based on the OWL-S upper ontology *ServiceProfile* section, *e.g.* [24] or the Web Service Modeling Ontology (WSMO) *Capability* descriptions, *e.g.* [25]. We, however, are dealing with the discovery of *workflow fragments*, and the difference between atomic services and workflows indeed makes a difference to the discovery task. In service discovery, *ServiceProfile* or *Capability* descriptions are used, which do not include control flow information and thus cannot be considered for workflow discovery purposes. Even though detailed control flow information clearly is present in OWL-S and WSMO ontologies through the *ServiceModel* and *Orchestration* descriptions, respectively, these parts of the ontologies are neither intended nor (to our knowledge) currently used to support discovery. We are now designing a workflow ontology which uses service orderings, conditionals and loops to represent and query workflow fragments. So far, based on OWL Lite (using `hasSuccessor`, `hasDirectSuccessor` and `partOf` roles, the last two of which are transitive), we can retrieve workflows based on Q1-Q2 for fragments in the Williams workflow (more detail can be found in [17]).

4.6 The Process Knowledge Acquisition Bottleneck

The question is then how to get annotations for workflows based on such a model. Scientists are reluctant to manually populate any model of an experiment. Techniques to address the process knowledge acquisition bottleneck are therefore needed. With respect to populating that part of the experimental model that supports repurposing, techniques from service ontology learning and automated service annotation are promising. 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. 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.

4.7 Lack of Workflow Fragment Rankings

Once workflows and annotations based on the workflow model are created, one can query these for relevant fragments (Q3). As workflow fragment discovery is about retrieving those fragments that are “close enough” to a user’s context, the notion of rankings and similarity 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 both developing suitable metrics for workflow similarity and generating rankings based on these metrics with query mechanisms.

Domain-dependent metrics relate processes on domain-specific issues. For instance, the choice of gene prediction fragments in Figure 2 depends on what species one is interested in. Given the evolutionary similarity between human and cattle, the prediction techniques used for these species (present in the Williams and Trypanosomiasis workflows) are more closely related to each other than to the techniques needed for the Aspergillus fungus. *Domain-independent metrics*, on the other hand, work over features such as data and control flow, calculating for instance how many services are to be moved, removed, added, replaced, merged or split to relate different fragments. This in effect would provide a measure of the *integration effort* involved to transform one piece of workflow into another.

In case one would like to produce rankings based on OWL ontologies, a mechanism will be needed to measure (dis-)similarity between fragment representations. For descriptions in OWL (Lite and DL), we need to retrieve those cases where two fragments are similar 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.

Three approaches have been proposed over the years to deal with the notion of similarity in DLs. The first is *feature-based*, and builds on the analogy of DL concepts and roles as pieces of conceptual knowledge, where some of the pieces (features) can be shuffled around. Feature-based approaches and implementations relying on structural algorithms have been developed for $\mathcal{FL}^-$ in using shared roles and role values for matching, and by counting shared parent concepts [26]. In [27] 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 [28]. This approach stays within the first-order logic paradigm. Two alternative approaches for similarity in DL bring in 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.* [29], whereas the *probability-based* approach tries to merge Bayesian inference with DL reasoning, *e.g.* [30]. The theory and practical implications of these alternative approaches are less understood.

We have tried to apply the feature-based approach for ranking fragments but, so far, have been unable to, given the expressive constructs used in our workflow ontology (details in [17]). If no abduction algorithm for OWL Lite can be devised, approximation [31] 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 Lite and devise query relaxation strategies for a query manager, treating the reasoner as a black box.

5 Conclusions

The vision for the Web of Science fits well with the vision for the Semantic Web [3]. We see reuse and repurposing as a way of bootstrapping the Web of Science by stimulating the dynamics of sharing and reuse of experimental components in the scientific community. In this paper we investigated what it would mean for scientific problem-solving knowledge, captured in workflows, to be found and adapted, *i.e.* repurposed. We presented evidence that e-Science is an area where workflows are already actively shared, reused and repurposed.

We identified seven bottlenecks for repurposing and related some of these to challenges for the Semantic Web. We considered whether two of the identified bottlenecks, workflow fragment discovery and the ranking of fragments, can be tackled by reasoning based on OWL. We found that the existing OWL-based service description frameworks and querying technology would need extending for doing so. In light of the evidence of reuse, we believe that e-Science offers an appealing test bed for further experiments with Semantic Web discovery technology.

Acknowledgements

This work is supported by the UK e-Science programme EPSRC GR/ R67743. The authors would like to acknowledge the ^{my}Grid team. Hannah Tipney developed the Williams' syndrome workflow and is supported by The Wellcome Foundation (G/R:1061183). We thank the survey interviewees for their contribution: Chris Wroe, Mark Greenwood and Peter Li (^{my}Grid), Ilkay Altintas (Kepler), Vasa Curcin (InforSense), Ian Wang (Triana), Colin Puleston (Geodise) and Ben Butchart (Sedna). Sean Bechhofer provided useful comments on an earlier draft.

References

1. T. Hey and A. Trefethen. The uk e-science core program and the grid. In *Int. Conf. on Computational Science*, volume 1, pages 3–21, 2002.
2. R. Stevens, H. Tipney, C. Wroe, et al. Exploring Williams Beuren Syndrome Using ^{my}Grid. *Bioinformatics*, 20:303–310, 2004.
3. J. Hendler. Science and the semantic web. *Science*, January 23 2003.
4. I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: the making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
5. F. van Harmelen. How the semantic web will change kr: challenges and opportunities for a new research agenda. *The Knowl. Eng. Review*, 17(1), 2002.
6. R. King, K. Whelan, F. Jones, et al. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971), 2004.
7. R. Hull, M. Benedikt, V. Christophides, and J. Su. E-services: a look behind the curtain. In *22nd Symposium on Principles of database systems PODS*, 2003.
8. 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.
9. D. Berardi, G. De Giacomo, M. Lenzerini, M. Mecella, and D. Calvanese. Synthesis of underspecified composite e-services based on automated reasoning. In *2nd Int. Conf. on Service Oriented Computing ICSOC*, pages 105–114, 2004.
10. A. Wombacher, P. Fankhauser, B. Mahleko, et al. Matchmaking for business processes based on choreographies. *Int. J. of Web Services*, 1(4), 2004.
11. E. Lee. Overview of the ptolemy project. Technical Memorandum UCB/ERL M03/25, University of California, Berkeley, July 2 2003.
12. Annette ten Teije, Frank van Harmelen, and Bob Wielinga. Configuration of web services as parametric design. In *EKAW'04*, 2004.

13. W. van der Aalst, A. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE TKDE*, 16(9):1128–1142, 2004.
14. P. Lord, S. Bechhofer, M. Wilkinson, et al. Applying semantic web services to bioinformatics: Experiences gained, lessons learnt. In *ISWC*, 2004.
15. E. Deelman, J. Blythe, Y. Gil, et al. Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing*, 1(1), 2003.
16. W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
17. A. Goderis, U. Sattler, and C. Goble. Applying descriptions logics for workflow reuse and repurposing. In *DL workshop 2005.*
18. S. Al Sairaf, F. S. Emmanouil, M. Ghanem, et al. The design of discovery net: Towards open grid services for knowledge discovery. *Int. J. of High Performance Computing Applications*, 2003.
19. 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.
20. S. Majithia, D. Walker, and W. Gray. Automated web service composition using semantic web technologies. In *Int.l Conf. on Autonomic Computing*, 2004.
21. I. Altintas, C. Berkley, E. Jaeger, et al. Kepler: An extensible system for design and execution of scientific workflows. In *16th Intl. Conf. on Scientific and Statistical Database Management(SSDBM)*, 2004.
22. P. Li, K. Hayward, C. Jennings, et al. Association of variations in I kappa B-epsilon with Graves' disease using classical methodologies and ^{my}Grid methodologies. In *UK e-Science All Hands Meeting*, 2004.
23. G. Joeris and O. Herzog. Managing evolving workflow specifications. In *3rd Int. Conf. on Cooperative Information Systems (CoopIS98)*, pages 310–319, 1998.
24. 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.
25. U. Keller, R. Lara, A. Polleres, et al. Wsmo web service discovery. WSML Working Draft D5.1 v0.1, University of Innsbruck, 2004.
26. S. Bechhofer and C. Goble. Classification Based Navigation and Retrieval for Picture Archives. In *IFIP WG2.6 Conference on Data Semantics, DS8*, 1999.
27. A. Cali, D. Calvanese, S. Colucci, et al. A description logic based approach for matching user profiles. In *DL workshop 2004.*
28. S. Colucci, T. Di Noia, E. Di Sciascio, et al. A uniform tableaux-based approach to concept abduction and contraction in aln. In *DL workshop 2004.*
29. C. Meghini, F. Sebastiani, U. Straccia, and C. Thanos. A model of ir based on a terminological logic. In *116th ACM SIGIR*, pages 298 – 307, 1993.
30. D. Koller, A. Levy, and A. Pfeffer. P-classic: A tractable probabilistic description logic. In *AAAI 1997*, pages 390–397, Rhode Island, August.
31. S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In *KR2002*, pages 203–214, San Francisco, USA, 2002.