Experiences with e-Science workflow specification and enactment in bioinformatics

Matthew Addis[1], Justin Ferris[1], Mark Greenwood[2], Peter Li[3], Darren Marvin[1], Tom Oinn[4], Anil Wipat[3]

[1][mja,jf,djm]@it-innovation.soton.ac.uk. IT Innovation Centre, University of Southampton, SO16 7NP
[2] markg@cs.man.ac.uk. Dept. of Computer Science, University of Manchester, M13 9PL
[3][Peter.Li, Anil.Wipat]@ncl.ac.uk. School of Computing Science, University of Newcastle upon Tyne, NE1 7RU
[4] tmo@ebi.ac.uk. European Bioinformatics Institute, Hinxton, Cambridge, CB10 1SD

Abstract

Workflow techniques form an important part of in-silico experimentation within the bioinformatics domain and potentially allow the eScientist to describe and enact their experimental processes in a structured, repeatable and verifiable way. Bioinformaticians routinely use Web-based resources within their in-silico experiments. However, the use of current web service orchestration techniques is problematic, and represents a significant barrier to take-up by the bioinformatics community, due to the rapidly evolving and competing standards, a lack of freely available tools, limited support for interaction with stateful services, and inappropriate levels of abstraction for the bioinformatics domain. As a result, the EPSRC funded myGrid[11] project has, in collaboration with the European Bioinformatics Institute and the Human Genome Mapping Project, developed a graphical toolset and workflow enactor which uses its own high level representation of a process flow, including specification of processing units, data transfers and execution constraints.

Introduction

Bioinformaticians frequently use a combination of local applications and remote services for performing 'in-silico' experiments. These experiments are procedures using computer based information repositories and computational analysis adopted for testing hypotheses or to demonstrate known facts. In myGrid's case, the emphasis is on data intensive experiments requiring combinations of applications into workflows. However, there can be a significant difference between the level at which the scientist wants to think about their problem and the level at which it is necessary to implement a solution, for example the details of the necessary web services calls and the data links between them.

As a result, we have developed a workflow language that allows a range of abstraction levels so that users who want to interact with individual services and applications at a detailed level can still do so, whilst others can be relieved from the nitty-gritty and focus on higher-level processes. In this way, our approach allows the members of the bioinformatics community to define the abstractions that fit their way of working and assemble workflows appropriate to their in-silico experiments.

This paper presents the experiences and rationale that have lead myGrid, in collaboration with the European Bioinformatics Institute (EBI) and Human Genome Mapping Project (HGMP) to develop yet another workflow language (SCUFL: Simple Conceptual Unified Flow Language). Details are given of our flow language and the corresponding workflow enactment engine. Our workflow-authoring tool is available as open source through the Taverna project [3], as is the workflow enactment engine, available through the Freefluo project [2]. Both tools are designed to be flexible and extensible to other domains.

Finally, a case study is included to show how we use our approach to author and enact a real-world bioinformatics workflow using Web Services.

Who, what, when, where, how of bioinformatics workflows in myGrid

Workflows for in-silico experimentaion

MyGrid aims to assist the scientist with the development and execution of in-silico experiments. These experiments allow the scientist to investigate or verify a hypothesis that they may have about a particular problem or domain. Such in-silico experiments are, by their

very nature, hypothesis driven, ad-hoc and highly specialised to the particular problem they are associated with. For example, myGrid is now finishing its first full prototype using a case study for the examination of the genetics of Graves' disease [4]. The associated workflow is very specific to Graves' disease and is used to investigate what genes and loci are involved, to then determine which single nucleotide polymorphisms (SNPs) located in these genes are involved and finally to develop genotyping experiments to test the above hypotheses.

The use of myGrid to investigate Graves' Disease is just one example of the workflow lifecycle where workflows are typically assembled and tailored to the particular experiment; enacted using a combination of local applications and remote services; iterated and refined; and then recorded for provenance alongside the experimental results. The exploratory and ad-hoc nature of the work performed means that the user will often interact with the workflow whilst it is executing, for example to visualise and filter intermediate results, choose appropriate remote service providers or local tools, and generally to monitor and control execution.

Workflow functionality

Basic workflow requirements such as the need to support sequential and parallel flows, looping and conditionals, recursion and complex data types (not just int, float, string etc.) are treated as a 'given' in this paper and are common to most workflow languages, i.e. they are not particular to workflow in e-Science. Other functionality, for example the need for semantic annotation and discovery of workflows, the need to generate provenance information, and the need to support services that generate large volumes of data are very much germane to e-Science workflow and are discussed in more detail later in this paper.

Integration, integration, integration…

In providing a general-purpose environment for in-silico experimentation using workflows, myGrid will need to accommodate/integrate a vast range of resources in terms of data and applications. These resources may be within an organisation, for example in-house systems at a large pharmaceutical company or local tools developed within an academic research group, or they may be external services delivered by a public body or accessed across an extranet. The European Bioinformatics Institute [5] alone hosts over 50 tools and 40 databases. The problem of dealing with the heterogeneity of bioinformatics resources is not the subject of this paper since the problem of integration is a well-known problem in the bioinformatics field [6]. There are several issues that arise from a workflow perspective due to resource heterogeneity:

Using the right level of abstraction

Workflow users will typically want to use remote services at different levels of abstraction depending on what they want to do. Some users will want to interact intimately with a specific service to tweak parameters that determine the detailed nature of the results or to tune performance. Other users will wish to be abstracted from these details since they are more concerned with the overall orchestration of several services into a high-level flow and hence want detailed workflows and specific invocation methods to be 'wrapped' up and delivered in an easy to use form.

Integration of existing tools and services

There are several existing tools or services that provide integration functionality and the user will want to incorporate these within their workflows. These tools and services often have their own invocation and scripting mechanisms. Interaction with such tools and services is often stateful and scripts may be used to describe a series of activities that need to be performed. Furthermore, a variety of type systems are encountered (conceptual types, data formats, 'on-the-wire' types etc.) depending on the tools and services being used. The requirements for abstraction and different invocation models can be illustrated by considering the use of the Talisman [7] tool and SoapLab [8,10] services, both of which have been developed by the EBI and are used within myGrid workflows.

Talisman is a rapid application development tool and runtime environment for writing web based user interfaces. A wide variety of applications and data can be accessed through Talisman, for example, the EMBOSS[9] toolset and the ENSEMBL[12] database. Talisman is currently used by curators at the EBI for the annotation of Interpro and GO. Talisman is typically used with a Web-based user interface, but it can also execute XML scripts that describe a series of activities to perform. This is in contrast to the typical Web Services model where a set of separate operations would typically be provided, one for each application that can be invoked. Therefore, incorporation of Talisman as a Web Service into a

workflow requires the use of XML scripts as well as XML data.

Soaplab is a set of Web Services that provide a programmatic access to applications on remote computers. The EBI has a Soaplab service running on top of several tens of analyses (most of them coming from EMBOSS). The advantage of Soaplab is the uniform way of describing analyses and their input and output data by an XML-based metadata description. Use of Soaplab requires a stateful interaction where a series of calls are required to execute an application (create instance of application, run application, wait for results, get results). Therefore, incorporation of Soaplab into a workflow requires a 'mini-workflow' to execute each application.

Workflows as part of 'e-Science'

Workflow lifecycle

Use of workflow as part of a scientific endeavour requires support for the workflow lifecycle. For example, a particular workflow will typically be authored, enacted, validated and modified in an iterative cycle. Whole workflows, or workflow fragments, will be published and shared so that others can use or learn from them, which in turn involves a process of annotation, discovery and personalisation. Therefore, workflow authoring, versioning, and scientific validation will be a key part of in-silico experimentation.

Semantic description of workflows

The workflows (and resources) for a particular in-silico experiment will not necessarily be known a-priori. Specification at a semantic level of the resources and activities required allows discovery of suitable resources and workflows in a way that is abstracted from the syntactic details of data formats or invocation mechanisms. The use of explicit and machine-readable semantics for the inputs, outputs, and function of a workflow increase the ability to share workflows since it allows workflows to be indexed, browsed, and searched according to their overall purpose rather than detailed syntax, data formats or service bindings.

Workflow provenance

Use of workflows as part of scientific activity often require provenance[16] data to be kept about the activities performed during workflow execution (recording of intermediate data sets, details of the specific service providers used, versions of data and tools involved, interventions were made by the user). Provenance support is needed in the workflow language (so that the required level of provenance can be specified); the systems used to enact the workflows (so that the specified provenance data is generated during execution); and data stores (so provenance data can be store d and subsequently retrieved).

Large datasets

Bioinformatics applications can generate large datasets. If these applications are executed as remote services and large datasets need to be transferred between these services, then, depending on network topography, it can be inefficient (and in some cases prohibitive) to transfer all the data to and from a workflow orchestration tool used to orchestrate the services. In particular, it doesn't make sense to unnecessarily transfer intermediate datasets to and from a workflow tool if some of the services are co-located at the same service provider. Options include streaming of data directly between services (or via intermediate repositories) or orchestrated set up of alterative network protocol transmission, e.g. sftp. Support for these models needs to be present in both the workflow language and workflow enactment engine since the control-flow is now separated from the physical data-flow. Many of the bio services in myGrid are hosted by the EBI and there is scope for enhancing these services to allow service-to-service data exchange, for example through the use of data caching EBI.

Deployment

Two modes of deployment are expected for the myGrid workflow editor and enactor. The first mode is where an individual uses the tools on their desktop (either directly, or within the myGrid workbench[14]) for orchestrating a set of remote services (and potentially local applications). To support this mode, the tools need to be freely available and easy to install and use. The bioinformatics community is a significant user of open-source. Many members simply cannot afford, or are not willing to use, proprietary and commercial offerings. The second mode of deployment is where an existing community service provider (e.g. the EBI) or large commercial organisation (e.g. a pharmaceuticals company) wants to provide new services that allow its users (public, collaborators or internal employees) to compile and execute workflows and their resources. For example, a community service provider might host a workflow portal that allows users to search through a directory of workflows,

select one (or build their own), and then execute it at the service provider site. This has requirements of robustness and scalability that are less evident in the personal 'desktop' use of the tools.

Summary of workflow requirements in myGrid

Workflow in myGrid is characterised by the following requirements:

1) The workflow language should allow varying levels of abstraction to match the needs of each individual bioinformatics user. This includes specification of resources or processes at a semantic level (conceptual type rather than invocation syntax or service 'instance' specification).
2) The workflow language and enactor should support the specification and generation of provenance data. This includes support for authoring and versioning, for example by annotating the workflow specification.
3) Workflow enactment should allow invocation of services using mechanisms other than simple Web Service calls, in particular stateful interactions with services and use of scripts.
4) Specification and enactment of workflows that involve sequential and parallel flows, iterations and conditionals, complex data types etc. – the usual stuff supported by workflow languages, e.g. BPEL4WS[13].
5) Workflow editing and enactment tools need to be simple use by bioinformaticians in a desktop (unix or windows) environment. Workflow editing and enactment tools also need to be able to be deployed within services provided to the community by organisations such as the EBI, e.g. to allow users to discover, compose and execute workflows using resources at the EBI.
6) The workflow language and enactor should be able to accommodate services that exchange data directly between each other or via intermediate data repositories.

Choice of a workflow language

There are many existing standards for workflow. A good review of Process Modelling Languages is given on http://www.ebpml.org. The obvious question that myGrid has, and continues to face, is whether any of the existing standards are a suitable starting point for what the project wants to achieve. Some of the relevant standards include: Xlang, WSFL, and BPEL from Microsoft and IBM; ebXML from Oasis; XPDL from the workflow management coalition; UML extensions and EDOC from the OMG; and WSCI, which is under the umbrella of the W3C. It is not the purpose of this paper to review or summarise existing standards. However, some of the reasons why myGrid had not opted to use an existing Web Service orchestration language are given below.

Shifting sands

The current workflow standards are in flux; it is not obvious which one is best for myGrid. The major players (BEA, Microsoft, IBM etc.) are all involved in multiple 'standards' (BPEL, BPML, XPDL etc.) and multiple standardisation initiatives (W3C, OASIS etc.). Alliances come and go and standards are moving quickly (for example, WSFL/Xlang→BPEL1.0→BPEL1.1 only took about 18 months). It takes a significant amount of time and effort to effectively track workflow standards.

Availability of simple, free and high-quality tools

High quality and free tools (e.g. open source) are typically not available to support current standards. There are several commercial offerings, for example BPEL through IBM's WebSphere or Collaxa's BPEL server), however proprietary and costly solutions significantly limit the target audience of myGrid. Furthermore, industrial strength solutions, e.g. WebSphere, are typically not easy to deploy on the desktop. Many users want a simple desktop tool that they can download, install and use with the minimum of support.

Levels of abstraction

Web Service standards such as BPEL don't have the levels of abstraction necessary for most bioinformaticians. For example, BPEL can't conceptually group together the operations involved when accessing an application through SoapLab in a way that is easily encapsulated and partitioned within a larger workflow.

Semantics

Web Service orchestration languages don't support specification of processes or resources at a semantic level since they are written directly in terms of the syntax of XML data and WSDL operations. Furthermore, since there is little hope of myGrid influencing these languages, there is not much scope for adding this support to the language either.

Workflow is not just for Web Services

Use of multiple invocation methods (CORBA, Web Services, Grid Services, local libraries) is typically not supported in existing workflow languages, which are often targeted purely at Web Service invocation. Furthermore, existing standards and tools don't cater for service-to-service exchange of large volumes of data.

Provenance

Existing standards don't explicitly support provenance or authoring and versioning. Lack of support for provenance applies both to the language used to specify a workflow and an engine used to execute a workflow specification.

The myGrid approach

MyGrid did initially adopt the WSFL language as the basis of workflow specification and enactment. To start with, this proved to be a good choice. Although a freely available WSFL enactment engine was not available at the time, the implementation of an enactor to support the subset of WSFL required proved quite simple. Use of an existing specification also saved time when looking to get a working demonstrator going early in the project, which was essential in capturing user requirements.

However, as the project progressed and more sophisticated workflows needed to be supported, it became clear that WSFL was no longer suitable for many of the reasons listed above. Other languages were considered, but ultimately myGrid made the decision to develop its own language and enactor as the most cost effective way to achieve the research objectives of the project. We did consider layering our higher-level language on top of an existing lower-level third-party language and tool-set, however, whilst this would potentially allow a degree of 'plug-and-play' of third-party software, it was felt that the extra effort of taking this approach did not justify the benefits.

The decision of myGrid to design our own language and tools from scratch has meant that myGrid can more easily investigate some of key research aspects of e-Science workflows in bioinformatics, for example what it means to add semantics to a workflow language, and how to specify and generate provenance information.

Overall, we feel that decoupling myGrid from the current turmoil of the workflow standards world means that the project can get on with the research work of building e-Science tools that operate at the right level of abstraction, are open-source, and most importantly will be adopted by the community beyond the end of the project.

The Scufl language and workflow enactment engine

The Scufl language is a high-level conceptual workflow language and full details of Scufl can be found on the Taverna Open Source project site http://taverna.sourceforge.net. Only a short summary is presented here to assist with interpretation of the case study presented later. A Scufl definition consists of three main entities.

Processors:
A processor can be regarded as a function of some set of input data to a set of output data, where each function may have side effects on the execution environment that are not encapsulated within the input / output specification. Processors therefore contain ports, which are named uniquely within the scope of the processor, are defined as either input or output and may have a type assigned to them in some type scheme, but this is not currently defined within the Scufl language. Processors have a set of named input ports, a set of named output ports, a name within the scufl space, and a current execution status (initializing, waiting, running, or completed).

Data links:
A data link represents the consumption of some processor output by an input of some other processor. In fact, there is nothing in the language to prevent a processor consuming one of its own outputs, although this may be rejected during the translation to some other format due to the implicit problems with cyclic workflows. Data links have a source processor and output port name, a sink processor and an input port name and an optional name within Scufl space.

Concurrency constraint:
Although the data link specifications are enough to ensure correct execution ordering, since we allow processors to have side effects on their environment it is often required to explicitly create constraints on the ordering of execution of different processors. Specifically, it is possible to create a gate constraint that must be satisfied before a processor can effect a particular state change; for example, processor one is only allowed to shift state from waiting to running when processor two has status 'completed'. Constraints have a processor controlled by the constraint, a

state change blocked in that processor, a gate condition, and an optional name within scufl space. Concurrency constraints are particularly useful in dealing with stateful interaction with services as shown below.

Workflow enactment engine

Details of the myGrid workflow enactment engine can be found on the Freefluo Open Source project site http://freefluo.sourceforge.net

Freefluo is a Java workflow orchestration tool for web services that currently supports a subset of WSFL as well as Scufl. Freefluo is very flexible and at its core is a reusable orchestration framework that is not tied to any workflow language or execution architecture. The enactor core supports an object model of a workflow in the form of a directed graph where each node has a state machine that defines its lifecycle. Workflow scheduling and state transitions are driven by message passing between nodes as execution of the workflow progresses. The core of the enactor is decoupled from both the textual form of a workflow specification and the details of service invocation and data model. This allows the core to orchestrate a workflow in a generic way.

The enactor core is used in the context of a particular language and service run-time environment. A workflow language parser is used to convert a textual workflow specification, e.g. a Scufl document, into the internal object representation of the enactor core. An invocation framework is then added to allow the enactor to actually invoke services in the run-time environment and deal with the specific data types passed between the services invoked, e.g. WSDL calls and XML message parts.

Freefluo can easily be extended to support different invocation methods (Web Services, Grid Services, CORBA) and has been used in other projects in this way, for example for using CORBA wrapped numerical methods and data sets in a steel modelling workbench currently in use by the European Coal and Steel Community.

It is the ability to extend the enactor's run-time that also enables easy integration of stateful services such as Soaplab since the 'mini-workflow' of using these services can be encapsulated in bespoke extension. Furthermore, the run-time extensions are a natural and simple place to provide features such as iteration over datasets and automatic type casting or conversion.

Freefluo also supports generation of provenance information (what, when and where for all activities performed in a workflow) and also provides service discovery via standard UDDI if a service is not bound in a workflow specification (soon to be supported in Scufl).

Graves' disease case studies

The aim of the Graves' disease scenario is to identify genes involved in Graves' disease (GD) using a microarray approach (Fig. 1). GD is caused by the secretion of thyroid-stimulating autoantibodies by the lymphocytes of the immune system. These autoantibodies stimulate thyroid cells via the thyroid stimulating hormone receptor and override the normal feedback mechanism, leading to hyperthyroidism [15].

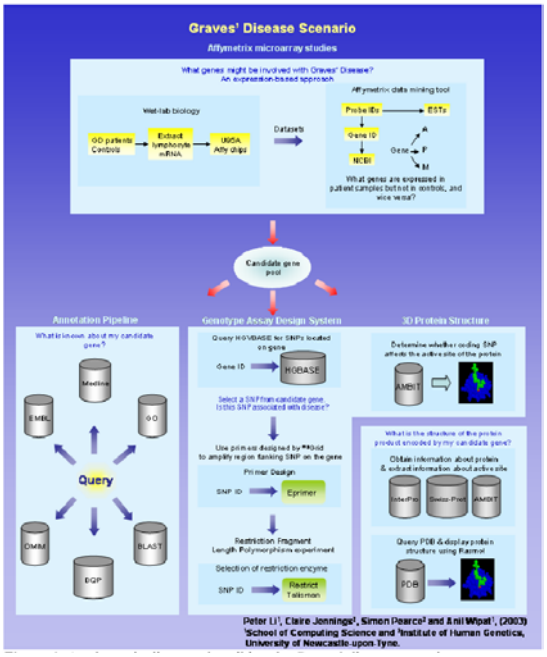

Figure 1 A schematic diagram describing the Graves' disease scenario.

High-level views of the workflows required for the GD scenario were represented using the unified modelling language (UML) in the form of activity diagrams (Fig. 2 represents just one sub-workflow within the overall GD workflow).

These series of conceptual steps identified for a workflow were then used as the basis of construction of a Scufl workflow specification using the Taverna workflow-authoring tool. The Scufl specification describes how to orchestrate the set of Web Services that provide the required

functionality. The workflow created is shown in Fig 3.

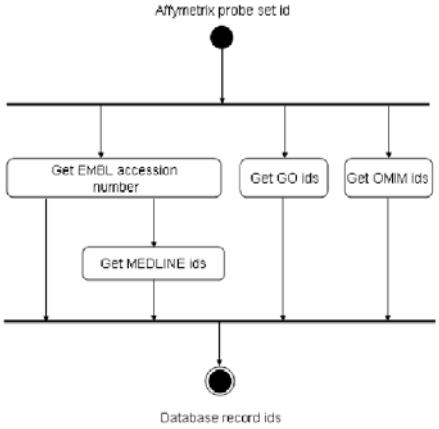

Figure 2 UML diagram representing the nucleotide sequence annotation workflow.

The triangles at the top of Fig 3 are workflow inputs, the triangles at the bottom are workflow outputs and the green ovals are Web Service operations. The solid lines represent the data flows with the text annotations showing the data types.

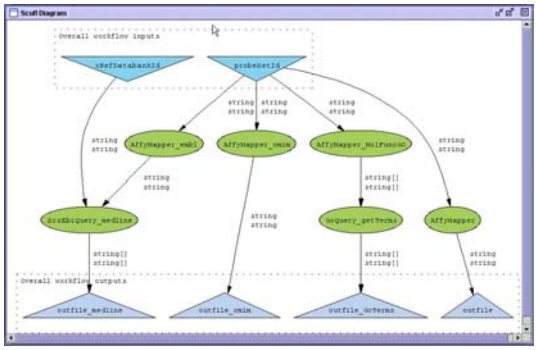

Figure 3 Graphical display of the 'Nucleotide Sequence With GO Terms' workflow.

Some parts of the Graves' disease workflow required the use of both the Talisman and Soaplab applications. An example of this is shown in Fig 4 where services are colour coded according to type: green for WSDL; pink for Talisman and beige for Soaplab. The important thing to note is that these all appear at the same level of abstraction to the user despite the different levels of complexity of invoking the Web Services involved. Also shown in the diagram is a series of Web Service invocations that are cascaded together using control links. This set of invocations corresponds to the use of a stateful Web Service where a series of calls needs to be made in order to execute the application and retrieve the results. In this case, the series of invocations is explicitly visible to the user, as they are not abstracted through an extension to the workflow enactor. It should be clear from this example how the workbench and enactor could be used to invoke other stateful applications, e.g. using CORBA or Grid Services.

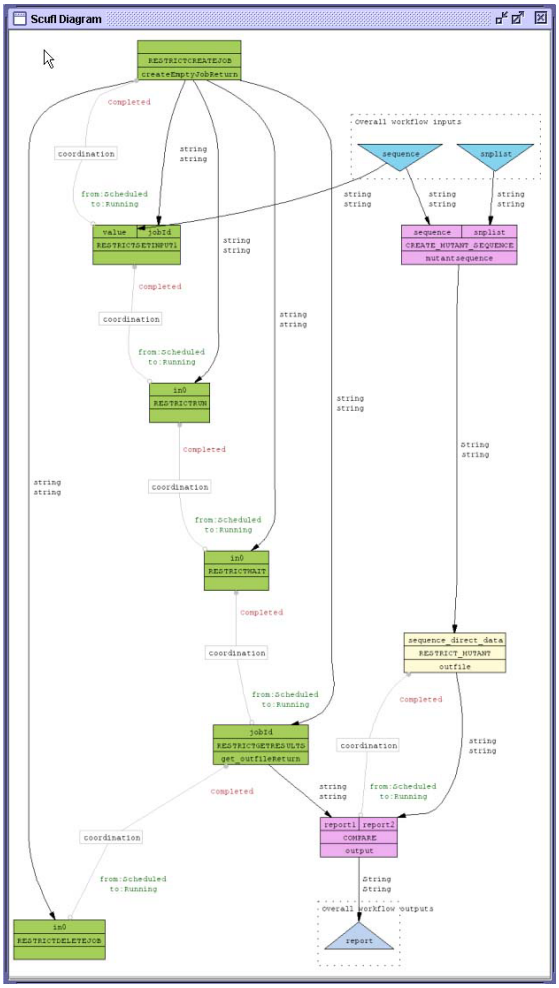

Figure 4 Incorporation of different types of service into a workflow

Next Steps

Whilst the work we have done to date has made good progress on Web Service orchestration, and fulfils many of the requirements of the developers and users of bioinformatics workflows, there are many key areas that we still need to address.

Future work includes: the explicit support for workflow semantics in the language and enactor, the ability to coordinate web services or other applications that need to exchange large quantities of data (effectively workflow enactment that passes data by reference instead of by value); coordination of contextualised web services, for

example passing context in SOAP headers using standards such as WS-Context; workflow management and lifecycle, for example a framework for the storage, indexing, searching and retrieval of workflows; and support for services or applications that stream input or output, i.e. they don't conform to the current Web Service paradigm; and integration of local applications and toolkits in their native form.

These areas will form part of a user-driven programme of further work within myGrid and in collaboration with the Human Genome Mapping Project and other e-Science projects.

Conclusions

MyGrid has, in collaboration with the European Bioinformatics Institute and the Human Genome Mapping Project, developed a graphical toolset and workflow enactor, which uses its own high level representation of a process flow, including specification of processing units with data and execution constraints. The workflow toolset is available as open source and has also been integrated into the myGrid workbench where it was a key tool in assisting with the identification of genes involved in Graves' disease.

The biologists involved in the myGrid case study have positively received our approach confirming the need to address the significant difference between the level at which the scientist wants to think about their problem and the level at which it is necessary to implement a solution. We also welcome wider feedback, anything from a quick note commenting on our work to offers of collaboration will be gratefully received. In particular we're interested in hearing from anyone who thinks they might be able to apply and extend our tools in other domains.

Acknowledgements

This work is supported by the UK e-Science programme under grant EPSRC GR/R67743. We would also like to acknowledge the assistance of the whole myGrid consortium.

References

[1] M Greenwood, C Wroe, R Stevens, C Goble, M Addis. "Are bioinformaticians doing e-Business?", EuroWeb 2002, St Annes College, Oxford, UK. 17-18 December 2002 http://www1.bcs.org.uk/DocsRepository/03700/3782/greenwoo.htm

[2] Freefluo Workflow Enactor: http://freefluo.sourceforge.net

[3] Taverna workflow authoring environment: http://sourceforge.net/projects/taverna

[4] R Stevens, A Robinson, and C Goble myGrid: Personalised Bioinformatics on the Information Grid, in Proceedings of Intelligent Systems in Molecular Biology, Brisbane Australia, July 2003

[5] http://www.ebi.ac.uk/services/index.html

[6] Bioinformatics: bringing it all together, NATURE, VOL 419, 17 OCTOBER 2002, http://www.nature.com/nature.

[7] Talisman: http://www.ebi.ac.uk/talisman/index.html

[8] SoapLab: http://industry.ebi.ac.uk/soaplab/

[9] EMBOSS: http://www.uk.embnet.org/Software/EMBOSS/

[10] M. Senger, Soaplab - a unified Sesame door to analysis tools, to appear in Proc UK e-Science programme All Hands Conference, 2-4 Sept 2003, Nottingham, UK

[11] C.A. Goble, C.J. Wroe, R. Stevens and the myGrid consortium, The myGrid project: services, architecture and demonstrator. Proceedings UK OST e-Science 2nd All Hands Meeting 2003, Nottingham, UK 2-4 Sept, 2003 http://www.mygrid.org.uk

[12] ENSEMBL http://www.ebi.ac.uk/ensembl/index.html

[13] BPEL4WS Specification: Business Process Execution Language for Web Services Version 1.1 http://www-106.ibm.com/developerworks/library/ws-bpel/

[14] Robert Stevens, Kevin Glover, Chris Greenhalgh, Claire Jennings, Peter Li, Melana Radenkovic, Anil Wipat, Performing in silico Experiments on the Grid: A Users Perspective. Proceedings UK OST e-Science 2nd All Hands Meeting 2003, Nottingham, UK 2-4 Sept, 2003

[15] Kohn LD, Shimojo N, Kohno Y, Suzuki K. (2000) An animal model of Graves' disease: understanding the cause of autoimmune hyperthyroidism. Rev Endocr Metab Disord. 1:59-67.

[16] Mark Greenwood,Carole Goble, Robert Stevens, Jun Zhao, Matthew Addis, Darren Marvin, Luc Moreau, Tom Oinn , Provenance of e-Science Experiments - experience from Bioinformatics, Proceedings UK OST e-Science 2nd All Hands Meeting 2003, Nottingham, UK 2-4 Sept, 2003.