WorkWays: Interacting with scientific workflows.

Hoang Anh Nguyen, David Abramson
Research Computing Centre, The University of Queensland, Brisbane, Australia
{uqhngu36, david.abramson}@uq.edu.au

Timoleon Kiporous
Department of Engineering, Cambridge University, Cambridge, United Kingdom
tk291@eng.cam.ac.uk

Andrew Janke, Graham Galloway
Centre for Advanced Imaging, The University of Queensland, Brisbane, Australia
{andrew.janke, gramham.galloway}@cai.uq.edu.au

ABSTRACT
This paper presents WorkWays, a workflow-based science gateway that supports human-in-the-loop workflows. The computational steering capability of WorkWays has been used to solve a number of problems where it is useful for users to study intermediate results and steer the computation. Two of those use cases are discussed in this paper.

Categories and Subject Descriptors
D.2.2 [Design Tools and Techniques]: User Interfaces
D.2.6 [Programming Environments]: Interactive environments

General Terms
Human Factors, Theory.

Keywords
Scientific Workflow; Science Gateways; Interactive Workflow-based Science Gateways; Human-in-the-loop workflow.

1. INTRODUCTION
Science gateways are increasingly used to interface scientists with their computations and their data. A science gateway is a Web portal integrating community-specific set of tools, applications and data collections, which provide access to resources and services of distributed computing infrastructure [1]. Computations launched behind a gateway can be individual applications or complete workflows. In particular, workflow-based gateways are extensible because workflows can be used to incorporate multiple, otherwise independent packages into a pipeline of activities. This approach has been exploited by many science gateways such as CAMERA [2] – a Kepler workflow portal supporting microbial ecology research, the LEAD portal [3] – a BPEL workflow portal for atmospheric research, or Galaxy [4] – a portal running the Galaxy workflow engine.

Existing workflow-based science gateways typically execute in batch mode. The gateways are mainly used to set up parameters, execute a workflow and collect computational results afterwards. Interaction between users and an executing scientific workflow has not received a lot of attention. In our opinion, gateways could support complex interactions between the running workflow and users, which would facilitate monitoring and steering of an execution as it progresses. This is important since it is sometimes difficult to replace human decision-making with automated algorithms. We have built an interactive science gateway, called WorkWays, that supports human-in-the-loop workflows. WorkWays allows users to interact with a running workflow to monitor and steer the computation, thus they become part of the computation. This human-workflow interaction is enabled by a dynamic IO model, which allows data to be inserted into, or exported out of a running workflow [5].

We have used WorkWays to solve a number of problems where users wish to study intermediate results and steer the computation. Computational steering capability not only provides better insights into the computation, but also allows users to focus on more significant parameter combinations. The problems we have solved so far span multiple domains, and often require different interaction modes, and/or different data types to be visualized. As an example, brain-imaging workflows may deal with processing image files, and it is useful to allow users to visualize and interact with these images as they are processed. In another example, the Parallel Coordinates technique [6] is a common way to visualize high dimensional data. Parallel Coordinates has been used effectively in aerodynamic optimization design to examine correlations between a very large dimensional parameter space and performance metrics [6]. Although this technique has been used mainly to post-process the optimization results, it can also be used to steer an optimization at runtime [6]. While implementing these workflows, we have expanded the existing IO model to include complex, domain-specific data types and new interaction modes.

2. SYSTEM ARCHITECTURE
Kepler was chosen as the workflow engine for WorkWays. The main reason of using Kepler is because of its rich set of components that can perform various tasks in different domains and across multiple computing platforms.

WorkWays’ Web components are implemented as portlets, conforming to JSR 286 standard [7]. This means that Web UI components can be deployed in any existing portal that supports the same standard.

User authentication is performed with the Australian Access Federation [8]. This allows users from universities and institutions participating in the federation to access to WorkWays using their institutional authentication.

2.1 IOFramework
The WorkWays IO framework enables interaction between the running workflow and Web portal by allowing data to be inserted into, or exported out of an executing workflow. Dynamic IO can be achieved with special actors in Kepler, which perform IO operations. IO actors serve as data sources, and initiate the connections and export data to the User Interface (UI) clients (data sinks). The UI clients then display the data, acquire user inputs and transfer input back to the corresponding IO actors.

A client-server architecture is used for connections between IO actors and Web clients, as shown in Fig. 1. This allows the Web clients and IO actors (and thus workflows) to be deployed on different machines. Additionally, it is easier to manage
connections between data sources and sinks, e.g. how many Web clients can connect to an IO actor at the same time, or whether a Web client can connect to an IO actor, etc. Connections from the clients to the IO Server are created using the IO Library, which defines an abstract interface for creating connections to the server. This abstract interface enables the system to support different connection protocols without affecting the upstream classes i.e. IO clients.

![Figure 1: IO Framework](image1)

To simplify the creation of IO actors, we created a generic and configurable actor named IOActor. IOActor allows users to declare and generate a new actor with IO specified operations, as shown in Fig. 2. Users can then employ the newly generated IO actors to exchange data between the workflow and UI clients. The UI clients display data according to the specified IO operations. WorkWays currently provides Web version of the UI clients (IOPortlets) and are working on the desktop UI clients. From the actor definition, the users can choose to export data to specific type of UI client, or multiple clients simultaneously.

![Figure 2: Sample actor configuration and the generated actor.](image2)

### 2.2 Workflow Editor

WorkWays also provides a facility for creating and editing workflows. Providing a Web-based workflow editor is important since it enables users to interact with the workflow in a uniform Web environment. This editing capability is achieved by executing Kepler’s editor in a virtual machine (VM) in the cloud. When the user edits a workflow, WorkWays launches a VM bundled with Kepler and a VNC server. The desktop environment with Kepler running is streamed to a portlet, which is a Web-based VNC client. The workflow is saved back to WorkWays database after editing is complete. Currently, WorkWays deploys virtual machines on the NeCTAR cloud platform, which is an OpenStack based research cloud in Australia [9]. Public WorkWays images are available in NeCTAR for users to launch their own VMs.

### 2.3 Workflow Deployment

WorkWays uses two different approaches to execute workflows: a centralised and a decentralised approach. In the centralised approach, a headless Kepler server receives Kepler XML declaration files from clients and launches workflow execution in separate threads. A disadvantage of this approach is that the server can be overloaded when multiple workflows are executed at the same time. In the decentralized approach, each workflow is distributed to one VM and executed independently. This is implemented in combination with the workflow editing mechanism. The user can execute the workflow in the same VM where editing is performed. This implementation does not require a Kepler server on the VM, and each VM acts like a separate desktop running Kepler. A new VM will be launched when the user wants to edit or execute another workflow.

The decentralized approach is very flexible, because it allows workflows to be executed in any VMs that are based on the public WorkWays image. This means that users can create their own VMs from a public image, and users are not restricted to the set of tools available in the headless server.

On the other hand, the centralised approach overcomes security restrictions of some clusters, in which computational jobs cannot be submitted externally. Kepler servers are then deployed on the clusters’ head nodes, and executing workflows sent from the Web portal.

### 2.4 Workflow Execution

WorkWays uses Nimrod/K [10] to manage execution of parallel workflows. Nimrod/K is an extension of Kepler using the well-established Nimrod family toolkit. Specifically, it augments Kepler with a director, called the Nimrod/K director, implementing the Tagged Dataflow architecture to expose and manage parallelism in a workflow [10]. Nimrod/K provides actors that send jobs to the Nimrod/G tool, which distributes and executes jobs on a range of HPC platforms from clusters, to the Grid and the Cloud. The Nimrod family toolkit is pre-installed in the WorkWays virtual machines.

### 3. USE CASES

WorkWays has been used in a number of studies [5]. In this section, we introduce two recent use cases for MRI imaging and Aero-dynamics design optimization.

#### 3.1 CVL Symmetric Model Creation

All brains exhibit individual variation, these differences may be small for some structures but the level of variability is sufficient to drive the development of average models from a population of individuals representing the taxa rather than being biased to an individual.

A robust atlas generation process should ensure that the resulting model exhibits the average morphology and signal intensity of the input datasets. To achieve this, model generation consists of an initial linear registration followed by nonlinear registration to an internally evolving average image. During model generation, a robust averaging approach is used in which voxels in the individual brain images that deviate from the current mean by more than a set amount are down-weighted. This is a very effective technique to identify and remove artifacts that are not consistent between individual images.

In this case study, WorkWays implements a process for creating average symmetric models, built in a similar fashion to Janke et al. [11]. The process involves a progression of 22 recursive fits to an evolving model utilising initially linear followed by progressively more dense nonlinear fitting. In the initial fit a random subject is chosen from the group and all subjects are fitted to this individual. An initial group average is built. An average transformation is then determined and the inverse of this average is applied to the resulting model. This process is then repeated with subsequent fits. In each iteration, the target is the model from the previous step. This iterative strategy converges quickly on the
true mean position and structure of morphological features in the chosen population.

Creating symmetric models is a computing intensive and long running process. As an example, this fitting process took 3 weeks on a 50-core commodity cluster [11]. During this process, the scientists need to periodically check intermediate output to confirm the correctness of the execution. Since this is a progressive process, an erroneous model would cause the subsequent models to be incorrect, and thus the execution should be terminated. To provide a user-friendly environment for scientists to execute this process, WorkWays supports such examination of images.

![Symmetric atlas creation workflow](image)

**Figure 3:** Symmetric atlas creation workflow.

Fig. 3 shows the workflow that creates the symmetric brain atlas. The workflow parameters are not shown in this figure in order to save space. The workflow takes, as inputs, a set of MINC [12] files containing 3D MRI scan of the subjects being studied. Intermediate and final models are also in MINC format. Essentially, the workflow has two branches: the upper branch for processing linear fitting (linear fitting stage) and the bottom one for processing non-linear fitting (nonlinear fitting stage). The loop in this workflow reflects the iterations discussed above. Before the execution, the user needs to provide a **stage parameter**, which specifies the fitting type and iterations of the process. This parameter is used by the workflow to select the branch to be executed. As an example, a configuration with **stage parameter “lin,1,3”** causes the workflow to execute linear fitting branch once and non-linear fitting branch twice, one with low resolution and other one with medium resolution. Each fitting stage generates an intermediate model, which will be used in the following fitting stage. The workflow generates a final model once all the fitting stages specified in the **stage parameter** are processed.

![Execution of Symmetric Model Creation workflow](image)

**Figure 4:** Execution of Symmetric Model Creation workflow.

To address the needs to examine intermediate models, we integrated the BrainViewer library [13] into the IO framework. This allows users to visualize and interact with MINC files. BrainViewer is a Javascript library that allows for real time manipulation and analyses of neuro imaging data in standard Web technology [13]. This library supports a wide range of neuro-imaging data such as MINC, MNI object format, etc.

![Image](image)

**Fig. 4** shows the page created by WorkWays for this experiment, which contains two portlets, a WorkflowEditor portlet and an IOPortlet displaying the intermediate models from the execution. At the time this figure was taken, the final model was not generated, and it was yet to be shown. At each iteration, a new intermediate model is generated and presented to the user. This IOPortlet allows the user to visualise and examine the 3D brain images in the MINC file of the intermediate model and to decide whether to proceed with the execution or not.

### 3.2 2D Airfoil Optimization

In this case study, WorkWays is used to implement an interactive 2D airfoil optimization process, as described by Kipouros et al. [14]. In this experiment, the Free Form Deformation (FFD) technique [14] is used to create new candidate airfoil geometries, either by local or global modifications. The leading and trailing edges of the airfoil are kept unchanged to keep the chord length and angle of attack fixed, as shown in Fig.5. The movement of the four control points in the Cartesian space then defines the parameter space of the optimization. Xfoil [14] is then use to calculate the lift, drag and moment coefficients of each new design parameter.

![Image](image)

**Figure 5:** FFD with four control points ([14]).

It is useful for scientists and engineers to explore and identify the physical and behavioral relationships between design parameters and performance metrics. The knowledge from such analyses can be used to reduce the number of design variables, or reduce the range of some design parameters. This reduction in terms of the size of the design space helps to accelerate the design convergence. However, such analyses are not trivial since they involve multi-dimensional data analysis of the parameter space and objective function(s) space simultaneously. Kipouros et al. [6] propose a method to conduct such analyses, in which Parallel Coordinates representation is used to analyse a multi-objective optimization process of turbo-machinery compressor blades and support human interaction. Parallel Coordinates allows a domain expert to interact with the system to identify which design parameters reveal more interesting and realistic optimum design configurations. This can accelerate the search process.

![Image](image)

**Fig. 6** shows the workflow that implements the interactive 2D airfoil optimization. The **DefineSearchSpace** actor initiates the optimization by defining the domain of the search; in this case,
this domain is the combination of possible coordinates of the control points. The starting points will be selected from this domain by the SelectPointsActor, and will be sent to the optimization actor (Simplex). The optimization actor generates set of points that are sent for evaluation. This workflow combines FFD and Xfoil into one composite actor: Compute xfoil. This composite actor evaluates the points from the optimization actor and the results of this evaluation are used to decide the next generations of points. This cycle stops when convergence criteria specified in the optimization actor are met.

This workflow uses three IO actors: Output and Optimum actors for visualising temporary and optimum shape of the foil, and ParallelCoordinates for analyzing relationships between design and object space. They are all generated from the generic I0Actor.

Fig. 7 displays the page created by WorkWays for this experiment, however, the WorkflowEditorPortlet is not shown this figure to save space. The three IOPortlets corresponds to three I0Actors in the workflow. The two IOPortlets on the right display the temporary and optimum airfoil shapes while the parallel coordinates IOPortlet is shown on the left. This parallel coordinate plot has 9 dimensions, consisting of 4 Cartesian coordinates of the control points, and the evaluation result. In order to guide the optimization to a particular parameter space, the users choose a sub-region of each dimension. Once the user clicks the submit button, this information is sent back to the ParallelCoordinates actor. This triggers the DefineSearchSpaceActor to generate new set of starting points.

Figure 7: Execution of 2D airfoil workflow in WorkWays

4. FUTURE WORK

In this paper, we have described WorkWays, an interactive workflow-based science gateway and two human-in-the-loop workflows from different domains. The architecture of WorkWays has been discussed. We use Kepler as the workflow engine, and JSR 286 portlets for its Web components. We also discussed the IO framework that allows users to insert data into, or export data out of a continuously running workflow. We then used two use cases to illustrate the computational steering capability of WorkWays in different domains.

There are currently several improvements in progress. First, we will expand the object types currently supported by the IO framework. We aim to integrate ParaviewWeb, which supports a much wide range of data types. Second, we will improve the way IO operations are created. A GUI will be created for specifying IO operations, instead of the current text-based approach. Third, we will extend the type of clients beyond the Web. One of such client is the OptiPortal [15] tiled display wall, which will enable visualization of high-resolution data, which often requires very large screens.

ACKNOWLEDGMENTS

We acknowledge the Australian Mouse Brain Mapping Consortium (AMBMC) for the access to mouse imaging data.

REFERENCES