Workspace - a Scientific Workflow System for enabling Research Impact

D. Watkins[a], D. Thomas[a], L. Hetherton[a], M. Bolger[a], P.W. Cleary[a]

[a] *CSIRO Data61, Australia*
Email: Damien.Watkins@csiro.au

Abstract: Scientific Workflow Systems (SWSs) have become an essential platform in many areas of research. Compared to writing research software from scratch, they provide a more accessible way to acquire data, customise inputs and combine algorithms in order to produce research outputs that are reproducible. Today there are a number of SWSs such as Apache Taverna, Galaxy, Kepler, KNIME, Pegasus and CSIRO's Workspace. Depending on your definition you may also consider environments such as MATLAB or programming languages with dedicated scientific support, such as Python with its SciPy and NumPy libraries, to be SWSs. All address different subsets of requirements, but generally attempt to address at least the following four to varying degrees: 1) improving researcher productivity, 2) providing the ability to create reproducible workflows, 3) enabling collaboration between different research teams, and 4) providing some aspect of portability and interoperability - either between different sciences or computing environments, or both. Most SWSs provide a generic set of capabilities such as a graphical tool to construct, save, load, and edit workflows, and a set of well proven functions, such as file I/O, and an execution environment.

Workspace is an SWS that has been under continuous development at CSIRO since 2005. Its development is guided by four core themes; Analyse, Collaborate, Commercialise and Everywhere. Three of these map to the SWS requirements:

- Analyse - improving researcher productivity and enabling reproducibility by enabling a higher level of reuse and sharing of software components
- Collaborate - enabling collaboration between different research teams by providing a common platform and interoperability between otherwise incompatible software elements
- Everywhere - the ability to build multiplatform applications leveraging a range of computing and communication platforms and to support transdisciplinary knowledge integration

Workspace's fourth core theme of Commercialise is a key differentiator to most other SWSs and one not usually at the forefront of a researcher's mind. Workspace has been developed with a goal to shorten the path from research to impact, which in many cases takes the form of translating research into usable, robust, standalone tools for research, industry and commercial partners.

Workspace provides significant productivity gains with a learning curve suited to a broad range of users, without requiring software engineering expertise. This is aided by an intuitive user interface and comprehensive help system, to produce workflows and applications. Key features include:

- Easily extendible plugin architecture which allows individuals and teams to easily add their own data types, algorithms and user interface components into the framework to use and share with others. This feature has been used to expose a number of popular scientific libraries (such as OpenCV, PCL and VTK) and languages (such as Python, R and MATLAB) in the framework.
- Flexible and powerful execution system enabling continuous, inline interaction with data throughout the workflow while it is running, facilitating rapid prototyping. Parallel and distributed execution of workflows is supported as are other execution models such as batch runs from a command line or embedding workflows as shared library calls from within other software.
- Highly optimised and customisable real time interactive 2D plotting and 3D visualisation, combined with strong point cloud, surface and volume geometry pre and post processing capabilities.
- Capability to package workflows along with plugins and user interfaces as distributable, standalone software applications. This represents a fast and cost-effective path to the commercialisation of research outputs.

This paper discusses SWSs and their requirements and then introduces Workspace. Finally a high level comparison of the more popular platforms is given.

Keywords: *Workspace, workflow, plug-in, visualisation, commercialisation, rapid prototyping*

1. INTRODUCTION

Scientific Workflow Systems (SWSs) have become an essential platform in many areas of research. "A scientific workflow system is a specialized form of a workflow management system designed specifically to compose and execute a series of computational or data manipulation steps, or workflow, in a scientific application" (Wikipedia, 2017).

As defined by Wikipedia (2017) specialised SWSs "provide a visual programming front end enabling users to easily construct their applications as a visual graph by connecting nodes together, and tools have also been developed to build such applications in a platform-independent manner (Johnson et al. 2009). Each directed edge in the graph of a workflow typically represents a connection from the output of one application to the input of the next. A sequence of such edges may be called a pipeline". This structure intrinsically provides a framework for both increased re-use of operations within workflows and distributed development and hence collaboration. Wikipedia (2017) elaborates that "Distributed scientists can collaborate on conducting large scale scientific experiments and knowledge discovery applications using distributed systems of computing resources, data sets, and devices. Scientific workflow systems play an important role in enabling this vision".

Key differentiators from traditional workflow processes as given by (Wikipedia, 2017) include:

- "providing an easy-to-use environment for individual application scientists themselves to create their own workflows
- providing interactive tools for the scientists enabling them to execute their workflows and view their results in real-time
- simplifying the process of sharing and reusing workflows between the scientists.
- enabling scientists to track the provenance of the workflow execution results and the workflow creation steps."

In discussing SWSs one issue is to determine the domain boundaries of what constitutes an SWS. Wikipedia (2017) states: "The simplest computerized scientific workflows are scripts that call in data, programs, and other inputs and produce outputs that might include visualizations and analytical results. These may be implemented in programs such as R or MATLAB, or using a scripting language such as Python or Perl with a command-line interface". Some scripting languages also have dedicated scientific support, such as Python with its SciPy and NumPy libraries. It is important to note that almost any scripting or programming language could be regarded as an SWS in some form, usually with the addition of some libraries or associated packages. However, it is valid to consider both the simplicity of installation and ease of use to be key features of a good SWS. In this paper we only consider SWSs that meet these criteria while acknowledging that this is largely a subjective decision.

Compared to writing research software from scratch, an SWS provides a more accessible way to acquire data, customise inputs and combine algorithms in order to produce research outputs that are reproducible. Today there are a variety of SWSs such as Taverna, Galaxy, Kepler, KNIME (Konstanz Information Miner), Pegasus and Workspace. Depending on the preferred definition this may also include software such as R or MATLAB. All address different subsets of user requirements, but generally attempt to address at least the following three:

1) improving researcher productivity and providing the ability to create reproducible workflows,
2) enabling collaboration between different research teams, and
3) providing portability and interoperability - both between different sciences as well as different hardware platforms and software environments.

To meet the above requirements SWSs typically provide a generic subset of capabilities such as a graphical tool to construct, save, load, and edit workflows, and a set of well proven functions such as file I/O, visualisation, database access, support for calling external functions and distributed/parallel execution. There is a broad range of other functionality and services that can be provided by an SWS. Most support a reasonable subset of these but the distribution of features and the depth of these capabilities varies from platform to platform. Each typically has a number of distinguishing features and particular strengths that may relate to specific scientific domains or deployment options. The choice of which particular SWS to adopt or whether to remain with a more traditional approach will often depend on the degree of match of the SWSs to the specific user needs. Given there are a number of criteria on which to distinguish SWSs, a number of papers describing selection criteria have emerged over the last decade or so (Gil 2007, Deelman 2009). In this paper, we provide a summary of one workflow platform, namely Workspace and present a high level comparison with other well-known SWSs.

2. WORKSPACE

2.1. Workspace history and key attributes

Workspace (Workspace, 2014) is an SWS that has been under continuous development at CSIRO (Commonwealth Scientific and Industrial Research Organisation) since 2005 and publically released in 2014. Its development is guided by four core themes; Analyse, Collaborate, Commercialise and Everywhere (Cleary et al. 2014; Bolger et al. 2015). Three of these map to the aforementioned SWS requirements in the following way:

- Analyse - improving researcher productivity and enabling reproducibility by enabling a higher level of reuse and sharing of software components
- Collaborate - enabling collaboration between different research teams by providing a common platform and interoperability between otherwise incompatible software elements
- Everywhere – the ability to build multiplatform applications leveraging a range of computing and communication platforms and to support transdisciplinary knowledge integration

Workspace's fourth core theme of Commercialise is one of the key differentiator to most other SWSs and one not usually at the forefront of a researcher's mind but which has become increasingly important for universities, corporations and research organisations. Since its inception, Workspace has been developed with a goal to shorten the path from research to impact, which in many cases takes the form of translating research into usable, robust tools for industry and commercial partners. For this reason Workspace has been developed to not only support the research process like other SWSs, but to then provide a simple and cost effective pathway to commercial quality software. The use of Workspace for providing a low cost development and translation pipeline for IP (Intellectual Property) is discussed in detail in Cleary et al. (2017) and Bolger et al. (2016) with case studies of the development of applications given by Murphy et al, (2014, 2017) and Cohen et al. (2017). Example uses of Workspace include fire modelling (Sullivan et al. 2013, Hilton et al. 2015, and Miller et al. 2015) and in supporting simulation and computational modelling (Cleary et al. 2015).

Workspace provides significant productivity gains for researchers with its accessible learning curve (requiring little prior software development experience), aided by an intuitive user interface and comprehensive help system, enabling short timescales to produce workflows and applications. Workspace and its associated plugins are used in collaborative environments between teams of scientists and engineers spanning different disciplines and having various degrees of programming knowledge. Workspace has particular strengths in the following areas (Hilton 2015, Murphy 2015, 2017, Sullivan 2013):

- Easily extendible plugin architecture that allows individuals and teams to easily add their own data types, algorithms and user interface components into the framework to use and share with others. This has been used to expose a number of popular scientific libraries (such as OpenCV (Open Source Computer Vision Library), PCL (Point Cloud Library) and VTK (Visualization Toolkit)) and languages (such as Python, R and MATLAB) in the framework.
- Flexible and powerful execution system enabling continuous inline interaction with data inputs and outputs at any position in the workflow as it is running. Parallel and distributed execution of workflows is also supported as are other execution models such as batch runs from a command line or embedding workflows as shared library calls from other software.
- Highly optimised and customisable real time interactive 2D plotting and 3D visualisations.
- Strong support for point cloud, surface and volume geometry pre and post processing capabilities which are essential for data intensive applications such as model geometry construction from LIDAR, laser and other surface scanning techniques and for input to computational modelling.
- Minimal computational overhead. Data is passed between operations in memory (where possible) to minimise I/O, and the engine and most operations are written in C++ and compiled to native code, to minimise processing overhead.
- Simple and intuitive drag-and-drop user interface, both for workflow design and connecting workflows to custom application user interfaces.
- The ability to interact with web services and other remote data sets.
- Pluggable scheduling architecture with built-in schedulers for PBS (Portable Batch System) and Slurm, as well as the ability to schedule workflows to the "Workspace Agent", an installable component on any Windows, Linux or Mac machine.
- Capability to package workflows along with plugins (such as the Point Cloud Plugin) and user interfaces as distributable, standalone software applications. The ability to package workflows as compiled distributable software products represents a fast and cost-effective path to the commercialisation of

research outputs, without the need to re-develop the modelling and visualisation software and without requiring end users to purchase and learn complex and expensive third-party visualisation packages.
- Support for provenance with ability to connect to/invoke user specified provenance engines.
- Database support with ability to connect via generic adaptor to most database engines.

2.2. Operations and constructing a workflow in the Workspace canvas

A Workspace *operation* represents a task or computation. It uses the data passed in through its inputs and it can provide results via its outputs. The inputs to an operation can either be set directly (via data type specific UI controls referred to as *widgets*) or can come from the outputs of other operations via connections. Each input and output of an operation has a strict data type and the *connections* between operations enforce that the types match or can be converted (e.g. an integer output can be converted by a connection to a string input). Figure 1 illustrates the key parts of an operation as seen in the Workspace GUI (Graphical User Interface):

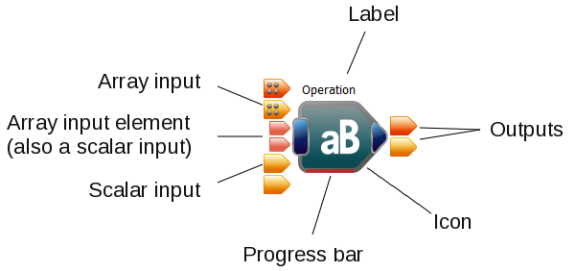

Figure 1. Operation visualisation showing its inputs and outputs.

Users drag instances of specific operation types onto the main canvas from an *Operation catalogue* and can set the instance's label to help self-document the workflow. They then create connections between operations by dragging between the inputs and outputs (the UI highlights compatible connections for the user to allow easy type matching, including between convertible types). When a user selects an operation on the Workspace canvas its inputs and outputs are displayed in the *Operation Editor* (Figure 2) allowing them to set values for inputs that are not passed to it from other operations. Users can use a context menu on an input or output on the operation to display the data with other available widgets for the given data type.

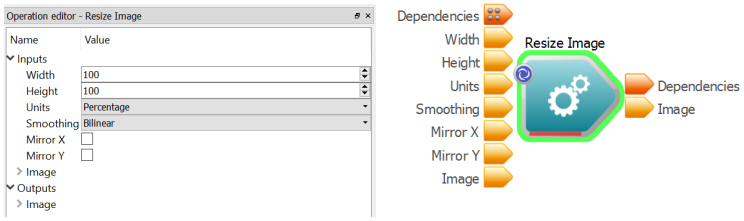

Figure 2. A "Resize Image" operation and the operation editor displaying its inputs and outputs

Once a network of operations and connections has been contructed the workflow can be executed. Workspace updates the network as a dependecy graph. To update or "execute" a given operation, Workspace first updates all inputs to the operation (which in the case of a connected input may require updating some "upstream" operations) and then executes the operation which performs its task and populates its outputs. When run in Workspace's interactive GUI the workflow will be brought up to date and will then wait for any changes. If any input then changes, Workspace will only re-execute the required parts of the network that was impacted by the change. UI widgets can be connected to any input or output throughout the network and these will update as their data changes, allowing users to see intermediate values at any point in the workflow. In additon to execution in Workspace's interactive GUI, workflows can also be run in batch mode, remotely scheduled to cluster job systems or embedded in other software via Workspace's C++ API (Application Programming Interface). A Workspace Python interface enables the creation, execution and monitoring of Workspace workflows from within a Python environment. Combined with a Python web server this interface enables Workspace workflows to be easily executed from within complex web applications.Workspace workflows can support complex flow control, loops and parallel execution. They can be nested to simplify representation of complex sub-workflows and can be externally referenced and reused by other workflows.

Table 1. High level comparison of major workflow platforms. ✓ indicates good support, – indicates that the functionality is not applicable or is unable to be adequately assessed, and x indicates the lack of the feature.

	Workspace	Taverna	Pegasus	KNIME	Kepler	MATLAB	Galaxy
Low barrier for non-expert developers	✓	✓	x	✓	✓	-	✓
Graphical workflow editor	✓	✓	x	✓	✓	x	✓
Cross-platform support (Windows, Linux, Mac)	✓	✓	x	✓	✓	✓	x
Tutorials, examples, documentation	✓	✓	✓	✓	✓	✓	✓
Concurrent (parallel/distributed/cloud) execution	✓	✓	✓	✓	✓	✓	-
Visualisation and input widgets available for complex types	✓	x	x	✓	✓	✓	✓
Multiple methods of execution available	✓	x	-	✓	✓	✓	-
Visualise data at any point in the workflow, while it is executing	✓	–	x	x	-	✓	-
Integrated advanced 3D scene rendering	✓	x	x	x	x	✓	x
Embed workflows into third party applications	✓	✓	-	x	-	✓	-
Support for mixing open and closed source components	✓	✓	-	✓	✓	-	x
Support for documentation of workflows	✓	✓	x	✓	✓	✓	✓
Supports calling external functions written in other programming languages and web services	✓	✓	✓	✓	✓	✓	✓
Ability to easily add datatypes	✓	x	x	x	✓	✓	✓
Ability to easily add operations	✓	✓	x	✓	✓	✓	✓
Ability to easily add widgets for visualising new datatypes	✓	x	x	✓	✓	-	-
Code wizards for adding new capabilities	✓	x	x	✓	x	-	x
Customisable User interfaces can be attached to workflows	✓	x	x	x	x	✓	x
Standalone application from workflow and UI	✓	x	✓	✓	x	✓	✓
Nested workflows (embedded, external or remotely defined)	✓	✓	✓	✓	✓	✓	-
Support for debugging and profiling	✓	x	✓	✓	✓	-	✓
Data and Process Provenance	✓	✓	✓	-	✓	-	✓
Support for IoT	✓	–	-	x	-	-	x
Support for mobile devices	x	x	x	✓	x	✓	✓
Interactive real-time workflows with streaming data	✓	x	x	x	-	✓	x
Low overhead in executing workflow	✓	x	-	✓	-	-	✓

2.3. Simple workflow construction example

Figure 3 shows a very simple workflow which reads an image from disk and resizes it. From left to right, an "Image Reader" operation has a file name input which could be set by the user using Workspace's file browser widget (which can be used on this type of input), or by any other string-adaptable widget. The output of this operation when it executes is an "Image" (of type QImage from Qt, one of Workspace's underlying libraries). This output is connected to the "Image" input of a "Resize Image" operation whose other inputs set the parameters of the resize process. The resize operation outputs the resized image (also of type QImage) which

is then passed out via the "Workspace Output" operation on the far right. When asked to execute a workflow, Workspace will attempt to bring all top level outputs (represented by this far right operation) up to date by working its way back through the relevant upstream.

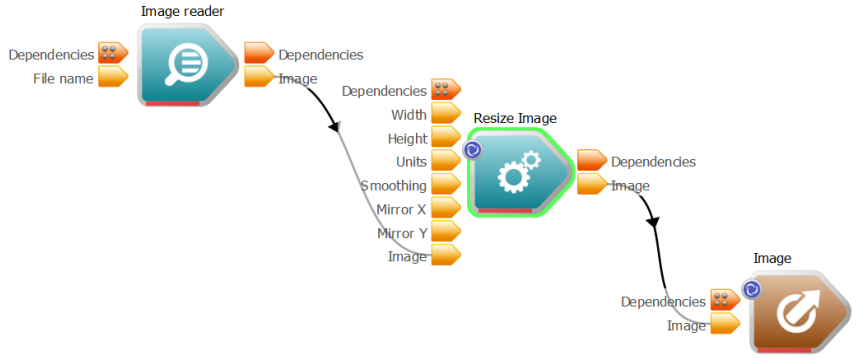

Figure 3. Simple workflow for resizing an image.

3. COMPARISON BETWEEN WORKFLOW SYSTEMS

It is useful to make a high level comparison of Workspace with other SWSs from a more qualitative perspective of whether a typical end user can readily make use of a specific functionality provided by the SWS. The functionality assessments are given in Table 1. It should be noted that the criteria chosen is not a complete list of all possible functionality a SWS can provide. The principal applied to this evaluation was "can an average user easily make use of specific functionality"- while it could be argued that almost anything is implementable in software by a knowledgeable software engineer, this was not the test applied with this evaluation. The criteria also reflects a range of more traditional requirements (such as a graphical workflow editor),emerging requirements such as support for mobile devices, Cloud and IoT, and with the four generic themes from the Abstract being borne in mind.. The functionality for each listed attribute is indicated by a tick if the platform is deemed to provide a good level of support and a cross if it does not. For cases for which a functionality is not applicable or not able to be assessed then no indication is given and the table entry is left blank. The authors acknowledge that such a rating is subjective.

It is evident from the table that all the platforms considered were found to have a broad range of functionality. It is arguable whether MATLAB should be considered as an SWS but its wide usage means that inclusion in the comparison is desirable. In terms of capabilities it ticks a broad number of SWS attributes. Taverna being one of the earlier developed SWS's appears to have less functionality in attributes relating to newer areas such as mobile devices and IoT. The strengths of Galaxy and Taverna also reflect the origins/uses of these platforms in bioinformatics. KNIME is a Data Analytics Engine and has seen adoption in a number of disciplines. Pegasus concentrates on distributed processing with provenance but lacks an easy entry for non-programmers as well as any graphical interfaces. Workspace compares favourably to the other platforms with good capabilities in all the identified attributes. Particular strengths include integrated 3D visualisation, code wizards for adding new capability, IoT and support for interactive real-time workflows including those dealing with streaming data. The ability to add customizable user interfaces is also a particular strength. This is a critical capability for supporting the translation of research IP at relatively low cost into user friendly application software as discussed by Cleary et al. (2017).

4. CONCLUSION

Scientific Workflow Systems (SWSs) are an increasingly popular solution to creating complex software applications with re-usable components developed in distributed and collaborative ways. They offer both flexibility at the development level and reduced cost by being able to more easily re-use components and to integrate with third party libraries and software. The characteristics and benefits of using an SWS were described.

The Workspace platform, being one such SWS, was examined in more detail and focused around four principles; Analyse, Collaborate, Everywhere and Commercialise. Strengths related to providing a low cost pipeline from development of experimental or prototype workflows to compiled distributable application (including GUI) were discussed. Readers were introduced to operations, their input and output definition and to connecting these to form connected graphs of operations in the form of workflows. Finally, a high level comparison was made between several of the more commonly known SWSs so as to provide a basis for users

to make informed choices around suitability for their needs. Workspace compares favourably with other platforms particularly with regard to supporting the translation of research IP.

5. ACKNOWLEDGEMENTS

The authors wish to thank Simon Harrison, Philippe Moncuquet and Chris Rucinski for their assistance with the assessment of various SWSs.

REFERENCES

Bolger, B., Cleary, P., Hetherton, L., Rucinski, C., Thomas, D., and Watkins, D. (2015). Workspace: Scientific Workflows and Applications for multiple Environments, Proc. eResearch Australasia Conference, Brisbane, Australia, 19-23th October.

Bolger, M., Cleary, P. W., Cohen, R., Harrison, S. M., Hetherton, L., Rucinski, C., Sankaranarayanan, N., Thomas, D., Watkins, D., and Zhang, Z. (2016). Workspace: a fast and low cost methodology for delivering commercial applications based on Research IP, Proc. eResearch Australasia 2016, Melbourne, Australia, 10-14 October.

Cleary, P. W. (2004). Large scale industrial DEM modelling, *Engineering Computations*, 21, 169-204.

Cleary, P., Bolger, B., Hetherton, L., Rucinski, C., Thomas, D., Watkins, D. (2014). Workspace: A Platform for Delivering Scientific Applications", Proc. eResearch 2014, Melbourne, Australia, 27-31 October.

Cleary, P.W., Thomas, D., Bolger, M., Hetherton, L., Rucinski, C., and Watkins, D. (2015). Using Workspace to automate workflow processes for modelling and simulation in engineering, MODSIM 2015, Gold Coast, Australia, December 2015.

Cleary, P.W., Watkins, D., Hetherton, L., Bolger, M. and Thomas, D. (2017). Opportunities for workflow tools to improve translation of research into impact, MODSIM 2017, Hobart, Australia, December 2017.

Cohen, R. C. Z., Harrison, S. M., and Cleary P. W. (2017). Dive Mechanic: Bringing 3D virtual experimentation to elite level diving using the Workspace workflow engine, MODSIM 2017, Hobart, Australia, December (2017).

Deelman, E., Gannon, D., Shields, M., and Taylor, I. (2009). Workflows and e-Science: An Overview of Workflow System Features and Capabilities, Future Generation Computer Systems, May, 2009, Volume 25, Number 5, ISSN 0167-739X, Pages 528—540, URL, http://dx.doi.org/10.1016/j.future.2008.06.012, DOI 10.1016/j.future.2008.06.012.

Gil, Y., Deelman E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., and Myers, J. (2007). Examining the Challenges of Scientific Workflows, IEEE Computer, vol. 40, no. 12, pp. 24-32, December, 2007.

Hilton, J., Miller, C., Bolger, M., Hetherton, L., and Prakash, M. (2015). An Integrated Workflow Architecture for Natural Hazards, Analytics and Decision Support, in: Environmental Software Systems. Infrastructures, Services and Applications, *IFIP Advances in Information and Comm. Tech*., 448, 333-342.

Johnson, D., et al. (2009). A middleware independent Grid workflow builder for scientific applications, 5th IEEE International Conference on E-Science Workshops. IEEE: 86–91. doi:10.1109/ESCIW.2009.5407993.

Miller, C., Hilton J., Sullivan A. and Prakash M. (2015). SPARK–A Bushfire Spread Prediction Tool, R. Denzer et al. (Eds.): ISESS 2015, IFIP AICT 448, pp. 262–271.

Murphy, T., Thomas, D. (2014). A user-friendly predictive model of arc welding of aluminium, Proc. 4th IIW Welding Research & Collaboration Colloquium, Wollongong, Australia, 5-6 November 2014, pp. 47.

Murphy. A., Thomas, D. (2017). MODSIM 2017, Hobart, Australia, December 2017.

Sullivan, A., Gould, J., Cruz, M., Rucinski, C., and Prakash, M. (2013). Amicus: A national fire behaviour knowledge base for enhanced information management and better decision making, 20th International Congress on Modelling and Simulation, Adelaide, Australia, 1–6 December 2013.

Wikipedia. (2017). https://en.wikipedia.org/wiki/Scientific_workflow_system

Workspace. (2014). DOI: http://dx.doi.org/10.4225/08/54D03170101B7