Workspace – a Scientific Workflow System with commercial impact

N. Oakes, L. Hetherton, M. Bolger, D. Thomas, C. Rucinski, D. Watkins and P.W. Cleary

CSIRO Data61, Australia
Email: nerolie.oakes@data61.csiro.au

Abstract: Compared to writing software from scratch, Scientific Workflow Systems (SWSs) provide a more productive environment to automate the

1) acquiring of data,
2) combing of algorithms, and
3) the orchestration of pre- & post-processing, in order to develop reproducible workflows and their outputs.

Workspace is an SWS that has been under continuous development at CSIRO since 2005 and which addresses several key challenges faced by the research and commercial communities. A key differentiator between Workspace and other SWSs is its facilities to supporting commercialisation of IP.

In this paper, we present a short overview of Workspace and its defining features and then describe in more detail some of the newer useability improvements and functionality provided to assist teams of users when creating applications and systems during development and execution – i.e. features and functionality normally referred to as supporting "programming in the large".

Recent additions have been driven by users creating increasing complex systems: where workflows may contain many operations, applications may use many workflows, and individual workflows may be edited on by multiple people. The additions include:

1) visual miniaturisation of input, output and variable operations,
2) improved loop construction and control,
3) extended workflow comparison, and
4) workflow merge.

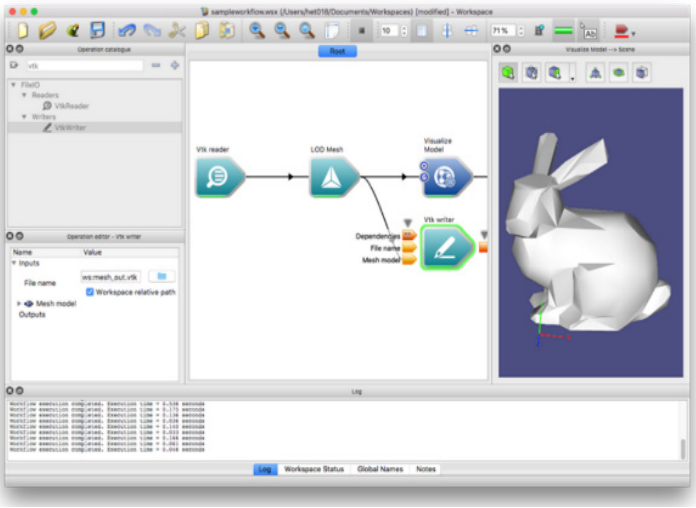

Figure 1. The workspace editor: an interactive workflow editing and execution application.

Keywords: *Workspace, workflow, visualisation, commercialisation*

1. INTRODUCTION

Scientific Workflow Systems (SWSs) are a key platform in many areas of research, providing a more accessible alternative to creating bespoke software. Wikipedia (Wikipedia 2018) describes SWSs as providing "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 process to the input of the next". This structure provides a framework for increased re-use within workflows and for distributed development and hence collaboration. Workspace. Critically, Workspace provides a number of features specifically designed to help customers develop commercial software that are not generally provided by SWSs. The key features of Workspace have been described by Watkins et al. (2017).

Workspace development is guided by four core themes: Analyse, Collaborate, Commercialise and Everywhere. Workspace was released for free public usage in 2014 (Cleary et al 2014; Watkins et al 2017). A key usage case for Workspace is in providing a low-cost development and translation pipeline for IP (Cleary et al. 2017; Bolger et al. 2014). Several case studies of Workspace applications have been published (Murphy et al. 2014; Cohen et al. 2018; Potter et al. 2017).

Workspace provides a rich GUI desktop workflow editing platform, as well as workflow execution, distribution, comparison and merging tools. These have substantial impact on the ability of users to use the SWS in a commercial frame.

2. WORKSPACE OVERVIEW

2.1. Workspace workflows

A Workspace workflow is comprised of a number of connected operations, each of which represents a self-contained task or algorithm. Operations are comprised of a number of components (Fig. 2 shows these as represented by the Workspace canvas). On an operation's left-hand side are a number of inputs, each of which can be either a scalar input (capable of accepting a single connection) or an array input (capable of accepting multiple connections). On the right hand side of an operation are its outputs.. Inputs and outputs are shown only when an operation is selected in the GUI, however pin buttons allow users to keep them visible if desired. Inputs and outputs are each defined with an association to a strict data type.

Connections, which represent data flows in-memory, can be created between inputs and outputs using drag-and-drop, with requirements for data type compatibility. Datatypes can be anything from simple C++ primitives, through to C++ classes, such as those contained within third-party libraries and frameworks.

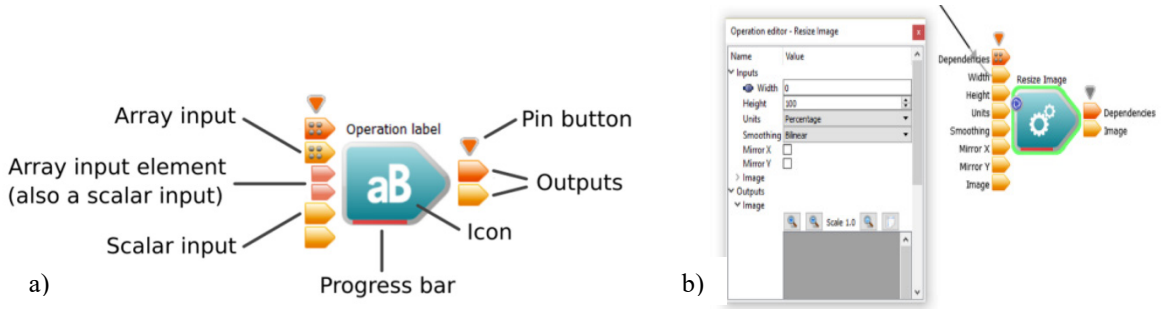

Figure 2. a) A Workspace operation as displayed in the Workspace editor, with specific elements identified **by name**, and b) An operation selected on the Workspace canvas. Note that the Operation Editor contains an editable field for each unconnected input, and connected inputs are explicitly marked.

2.2. The Workspace Editor

To create instances of operations, users drag-and-drop them from the operation catalogue onto the Workspace canvas. When the user selects an operation the operation editor is updated to allow the user to edit the values of unconnected inputs. The user can select an alternative widget to view/edit the data for a given input through the right-click context menu. There are several core operations and data types that are fundamental to the structure of Workspace workflows:

- TryInputs and SelectInput, operations affect dynamic flow. SelectInput dynamically selects which connected input to use based on the value of an "index" input. TryInputs instead tries each connected input branch in order, stopping as soon as one succeeds.
- ForLoop, ConditionalLoop, ParallelLoop and FileLoop, each of which facilitate repetition of a sequence of operations (these have recently been revamped to simplify their usage).
- The Workspace operation. A Workspace workflow can itself contain nested workflows. These make use of WorkspaceInput and WorkspaceOutput operations to define the data that they require and produce.
- ObjectArray, a serial grouping of data objects of any type can be constructed on-the-fly inside a workflow.
- ObjectDictionary, an associative container of strings mapped to data objects of any type. Hierarchies can be created by including other ObjectDictionary objects.
- ObjectGroup, the base-class of all composite data types. Members can be modified with ComposeGroup, DecomposeGroup and ModifyGroup, operations.

To execute a workflow a user first nominates one or more workspace outputs; a representation of the high-level data being produced by the workflow as a whole. Workspace works backwards from these outputs along the dependency graph formed by the operations and connections to identify the operations that need to be brought up-to-date. In this way, Workspace is able to a) avoid updating operations that do not contribute to the outputs being generated and b) avoid re-executing costly operations which are up-to-date.

If the user is executing the workflow in "continuous mode" (the default in the Workspace editor), once Workspace has brought all of its outputs up-to-date, it then monitors the workflow for changes and, once detected, immediately re-executes the required subset of the workflow. This mechanism underpins many advanced features, such as the ability to create fully-featured standalone applications (Cleary et al, 2019).

GUI widgets can be created to visualise data attached to any input or output in the workflow, including while it is executing. This allows users to observe intermediate results at any stage. Combined with the continuous execution mechanism described previously, users are able to rapidly prototype outputs with the effect of any modifications immediately observable without manual steps to refresh or regenerate the display.

The graphical workflow editor (workspace-gui) is only one mechanism enabling workflow execution:

- Workflows can be executed from the command line (such as on batch or cluster systems) using the workspace-batch executable, through which it is possible to pass data into the workflow via command-line arguments. Users expose inputs or outputs during workflow design by marking them as global names.
- Workspace's pluggable scheduling capability permits it to schedule workflows (such as instances of a nested workflow within a loop) to remote systems for execution. Out-of-the-box, Workspace provides schedulers for PBS, Slurm and the Workspace TCP-IP-based server.
- Workspace provides tools to allow users to create custom graphical user interfaces. Users build GUI components in Qt Designer (a standalone application developed by Qt and shipped with Workspace) and then use drag-and-drop to connect any input or output marked with a global name to a compatible widget (Fig. 3). The UI file can then either be opened and used inside the Workspace editor, or a code wizard can be used to autogenerate code necessary to combine the GUI definition and workflow into a standalone executable.
- Workspace provides a C and Python interface (CSIRO Python Online), which when combined with a web server, enable users to instantiate and execute workflows from within web services or web applications. As with the case of standalone executables, continuous execution mode can be utilised to maximise asynchronous processing.

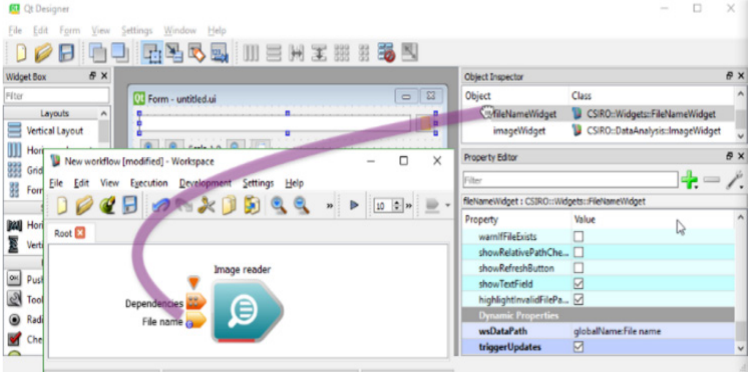

Figure 3. A user connecting an input to a widget in a custom UI using drag-and-drop from the Workspace editor to Qt Designer.

2.3. A simple workflow

Fig. 4 shows a simple but powerful workflow that reads surface mesh data from disk, reduces its level of detail (LOD) and then creates a visualisation of the results which is shown in Fig. 5.

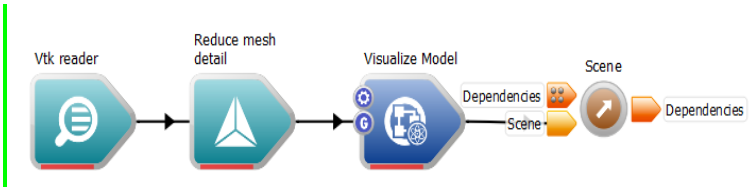

Figure 4. Example workflow for reducing the level of detail of a mesh and visualising the results.

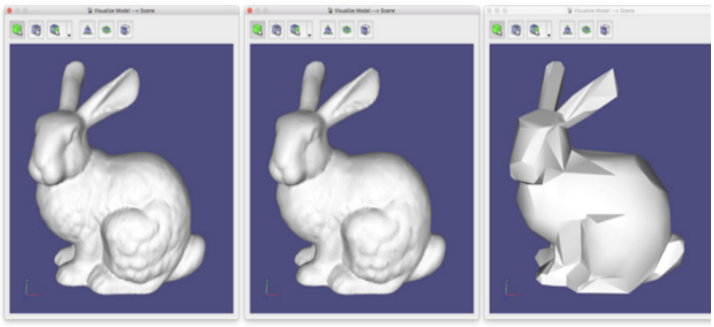

Figure 5. Example workflow results with (left) LOD reduction 0.0, (centre) results with moderate LOD reduction of 0.75, and (right) results with an extreme LOD reduction of 0.99.

3. RECENT CAPABILITY EXTENSIONS

Development of Workspace is ongoing, with priorities influenced by users' needs identified from feedback from a number of external users. In the last 18 months, the Workspace team has added new capabilities, including:

- The miniaturisation of inputs and outputs to reduce visual overload
- The simplification of the user interface for looping operations
- The extension workflow comparison capabilities
- The addition of a workflow merge functionality

Each of these capabilities is described in more detail in the sections below.

3.1. Miniaturisation of inputs and outputs

Workspace supports the concept of nested workflows. Nested workflows add a level of abstraction/structure to workflows and facilitate re-use. Sequences of operations can be grouped into a nested workflow and as such are represented as a single operation at higher levels. Users can then enter a nested workflow to see the actual workflow. Nested workflows can be added to the operation catalogue and subsequently when added to a workflow they can either be embedded (i.e. a copy of the workflow is inserted verbatim into that workflow) or the nested workflow can be referenced (in which case the external workflow file is used – meaning a single nested workflow can be used in many other workflows.)

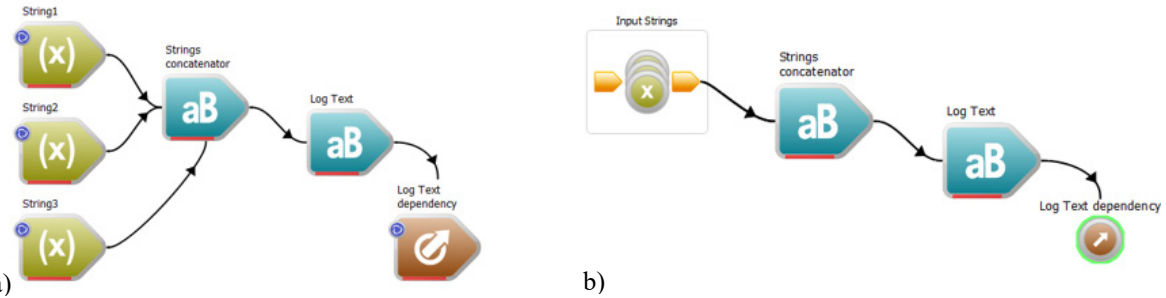

Figure 6. (left) a simple workflow in the old nested workflow format, and (right) the same workflow in new presentation, with the functional operations having a higher visual profile than the numerous IO and Variable operations.

When dealing with nested workflows the three most basic operations, namely WorkspaceInput, WorkspaceOutput and Variable, are used to display connections between workflows on different levels. When dealing with large complex workflows with many nested workflows, users have often reported that the proliferation of WorkspaceInput, WorkspaceOutput and Variable has added distracting clutter to the visual representation, reducing user productivity. To address this WorkspaceInput, WorkspaceOutput and Variables have had their visual profile reduced on the Workspace canvas in order to reduce their prominence. This increases the emphasis on the functionality of the workflow and allows better visual separation by users of inputs and structure. Further enhancing this, operations can now be faded or hidden entirely, or can be grouped into expandable racks that are only expanded if the user mouses over them or pins them open.

3.2. Improved Loop Construction and Control

Workspace supports the concept of looping (i.e. repetition of operations). Workspace has permitted loop bodies to either be contained in nested workflows or be represented as a sequence of operations at the same level as the loop itself. In response to user feedback that existing Workspace loops were confusing, a new set of simplified loop operations have been developed. A loop is now a visually-isolated nested Workspace that iterates multiple times until control inputs/outputs determine that it can stop. New loop operations include the ForLoop, ConditionalLoop, and FileLoop.

- A ForLoop iterates a fixed number of times, passing the current counter value into loop body. It is possible to iterate in parallel by checking the *Enable parallel looping* input
- A ConditionalLoop iterates until the *Keep looping?* WorkspaceOutput is set to false inside the loop
- A FileLoop loops over a set of files in a directory. The *Filter* and *Sorting* inputs control which files are included and in which order.

All existing loop operations remain, but are deprecated, allowing existing workflows to continue functioning and simplified loops to be phased in over time.

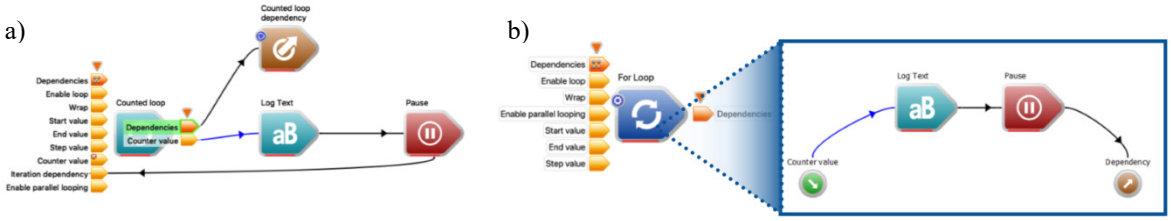

Figure 7. Simplified Workspace loop construct, a) old loop form, and b) new loop form (the break-out shows how the loop operation is now a nested Workflow, with the looped operations inside).

3.3. Workflow comparison

When multiple users/developers are developing a workflow or application, it is not uncommon that different users have to made changes to the same workflow. Generic text-based differencing tools are limited to identifying textually-sequential differences. Since Workspace workflows are represented in XML allowing significant variation in format and component ordering and then displaying in a graphical environment, workflow documents are often too complex for users to manually identify even with small functional differences, while generic tools are not tuned to identifying non-cosmetic differences. With a custom workflow-differencing tool a user can benefit from computer-aided difference-highlighting, and with a graphical interface they are able see easily what the key differences are. The challenges in developing such functionality included identifying categories of workflow differences (including making broad assessments of whether two differences could be considered as equivalent), assessing their relative importance and making the result visually clear.

Workspace-diff is a two-way comparison tool for Workspace workflows designed to simplify the identification of workflow differences. Users select two workflows, typically originally having an original parent that both are modified forms of. The display widget highlights elements that exist in one workflow and not the other, or that exist in both but are different in some way. Currently it will highlight difference to operations, connections, notes, inputs and outputs, and, optionally, display elements (which is a much broader range of content than dealt with by purely text-based diff tools). Elements that exist in one but not the other are shown in green where they exist or in red as “ghost” elements where they do not. Differences are shown in yellow.

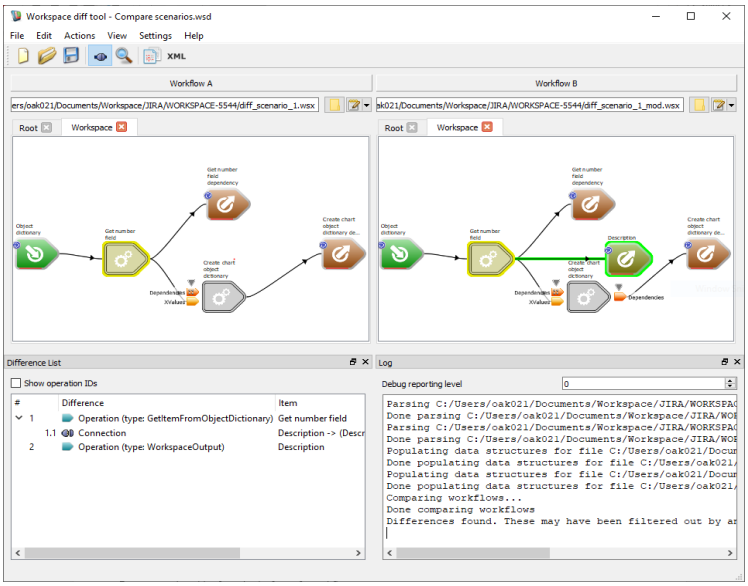

Figure 8. Workflow Comparison using the Workspace-diff application.

3.4. Workflow merging

The workflow merge tool is as an important extension of the workflow comparison tool. Where two versions of a base workflow exist, the user may want to selectively merge the two. While a comparison tool can identify such changes, often each version possesses unique or useful attributes which the user will want to preserve in the composite merged workflow. Workspace workflow documents are generally far too complex (because of the substantial non-text aspects of a workflow) for this to be done either using generic text-based file-merging tools or manually. With a custom, graphical workflow merging tool, the user can clearly see the differences between the base workflow and both of the variants. Challenges include identifying categories of changes, deciding whether changes in the two workflows are compatible, providing a visually clear display and ensuring that the result is always a valid workflow.

Workspace-merge is an automatic three-way merging tool for Workspace workflows; a high value capability for larger development teams that make heavy use of software version control systems such as Git or Subversion. The tool compares two workflows that have a common base (meaning that there is significant overlapping source to provide meaningful context for identifying discrete and actionable differences) and automatically merging them, highlighting and conflicts and supporting reconciling them where they occur. The display is colour-coded for source identification and has a user-defined set of allowed changes: the merge is rejected if there is no sensible way of handling it (or the user-settings have forbidden the auto-merge type).

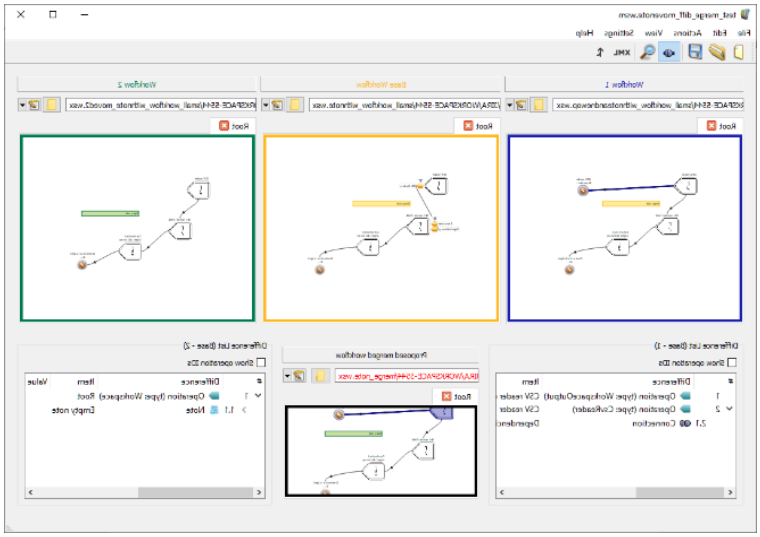

Figure 9. Example of the Workflow merge application.

4. CONCLUSIONS

Workspace is a workflow system that provides powerful capabilities to reuse components, algorithms and sub-workflows in different configurations and for different purposes. Workspace comes with a continually-evolving set of analysis tools that allow users to rapidly prototype both workflows and user-friendly applications, as well as debug, profile, validate and workflows. In this paper we presented recent Workspace developments. These include improvements to the workflow-creation process, a new set of easy-to-use looping operations. To improve use and interpretability of existing workflows, mini-operations and operations stacking have been introduced. Tools to diff, profile and debug have also been created.

REFERENCES

Bolger, M. *et al.* (2016) ,Workspace: a fast and low cost methodology for delivering commercial applications based on Research IP, *eResearch Australasia*, 6–10.

Bradski, G. (2000). The OpenCV Library, *Dr. Dobb's J. Softw. Tools*.

Car, N. J. (2013). A method and example system for managing provenance information in a heterogeneous process environment – a provenance architecture containing the Provenance Management System (PROMS), *20th Int. Congr. Modelling Simulation*, 824–830.

Cleary, P Bolger, M, Hetherton, L Rucinski,C Thomas,D and Watkins, D (2014). Workspace: A Platform for Delivering Scientific Applications, *eResearch Australasia*, 27–31.

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, *21st Int. Congr Modelling Simulation*, 669–675.

Cleary, P, Watkins, D., Hetherton, L., Bolger, M. and Thomas D. (2017). Opportunities for workflow tools to improve translation of research into impact, *22nd Int. Congr. Modelling Simulation*, 1–7.

Cleary, P. W., Thomas, D., Hetherton, L., Bolger, M., Hilton, J. E., and Watkins, D., (2019). Workspace: a workflow platform for supporting development and deployment of modelling and simulation*, revised for: Mathematics and Computers in Simulation*.

Cohen, R. C. Z., Harrison, S. Cleary, P. W., and Bolger, M. (2018). Dive Mechanic: Bringing 3D virtual experimentation to elite level diving using the Workspace workflow engine, *22nd Int. Congr. Modelling Simulation*, pp. 431–437.

GNU Lesser General Public License. [Online]. Available: https://www.gnu.org/copyleft/lesser.html.

CSIRO, Workspace. [Online]. Available: https://research.csiro.au/workspace/.

CSIRO, Workspace Python Interface. [Online]. Available: https://github.com/csiro-workspace/workspace-python.

Curcin, V. Ghanem, M. and Guo, Y. (2010). The design and implementation of a workflow analysis tool, *Philos. Trans. R. Soc., A*, vol. 368, no. 1926, 4193–4208.

Johnson, D., Meacham, K. and Kornmayer, H. (2009). A middleware independent Grid workflow builder for scientific applications, Proc. 5th IEEE Int. Conf. e-Science Workshops, 86–91.

Murphy, T. and Thomas, D. (2014). A user-friendly predictive model of arc welding of aluminium, 4th IIW Welding Res. & Collaboration Colloq., p. 47.

Potter, D.F., Khassapov, A., Pascual, R., Hetherton, L., Zhang, Z., (2017). Heliosim: A Workspace-driven application for the optimisation of solar thermal power plants, 2*2nd Int. Congr. on Modelling and Simulation*

Rusu, R. B. and Cousins, S. (2011) 3D is here: Point Cloud Library (PCL), IEEE Int. Conf. Robotics and Automation.

Watkins, D., Thomas D., Hetherton, L., Bolger, M. and Cleary, P.W., (2017). Workspace – a Scientific Workflow System for enabling Research Impact, *22nd Int. Congr. Modelling Simulation*.

Wikipedia, "Wikipedia page on Scientific Workflow Systems," 2018. [Online]. Available: https://bit.ly/2Ok1bZq.