Extending the capabilities of applications built using the Workspace architecture

R. Subramanian[a], D.G. Thomas[a] and S. J. Cummins[a]

[a] CSIRO Data61, Clayton VIC Australia
Email: Rajesh.Subramanian@data61.csiro.au

Abstract: Workspace (Cleary et. al (2020), Bolger et. al (2016), Cleary et. al (2013)) is a powerful, cross-platform scientific workflow framework that enables collaboration and software reuse. Workspace allows for the creation of custom operations that can be used in conjunction with a large set of powerful built-in operations to create scientific workflows. These custom operations can also be used in existing applications that are built with the Workspace architecture, enabling low-cost extension of the application capabilities without major reworking.

This paper describes the challenges a developer will face while producing a code that needs to be integrated into existing applications, and how the Workspace ecosystem can mitigate many of the obstacles in accomplishing this. This is illustrated with a case study, in which the following challenges are discussed:

1. Reusing the existing code that has already been built into an application.
2. Retrofitting the new algorithm code into existing applications.
3. Converting between the different data types that the old and new algorithms may work with.

Our specific problem:

We have a set of approximately twenty data processing utilities that allow analysis of specific types of simulation outputs, (see Cleary et. al (2020), Cohen et. al (2020) and Thomas et. al. (2019) for further details on these simulations). We would like to add general classes of additional analytical capability that can be deployed into these utilities (either into all of them, or on a user-need basis). Therefore, a flexible solution is required where code can be added or enabled across a broad range of similar utilities that have been developed by a range of different developers over several years.

In this specific case, we have developed a new Workspace operation that can perform smoothing of time-series data, and we would like to deploy this capability into some of the existing utilities.

The solution involved:

1. Adapting various kinds of data produced by the existing utilities to standardized types of data using the features available within the Workspace ecosystem.
2. Connecting the standardized data to the newly developed time-series smoothing operation and using it in lieu of the existing code.
3. Adding further plotting capabilities to assess the performance of the smoothing operation.

The existing applications and utilities that needed additional analytic capabilities were previously built on Workspace, which offered benefits in terms of re-use, extension, customization and commercial licensing. It is demonstrated that Workspace provides the flexibility and ease-of-use to allow the capabilities of a family of software applications to be effectively extended to make use of the new capability, which in this particular case are time-series smoothing algorithms. This is done without the need for any major rewriting or bespoke re-working of each software family member providing a large benefit for moderate cost.

Keywords: *Workspace, extensible architecture, plotting, scientific workflow software*

1. INTRODUCTION

Computational modelling and simulations usually involve a combination of several steps like reading and processing inputs, performing computation, producing various kinds of outputs, analysis and conversion of data to name a few. Some of these steps may need to be performed manually, which could be time-consuming, error-prone and expensive. Other steps may need to be performed using various software tools that may not be well documented. These factors create unique challenges in terms of provenance, reproducibility, and the ability to transfer the said simulation to another user.

A Scientific Workflow System (SWS) provides a more convenient way to perform these steps in the form of a workflow that can be saved and reused. Workspace (Cleary et. al (2020), Bolger et. al (2016) and Cleary et. al (2013)) is an SWS that has been under continuous development at CSIRO since 2005. Workspace attempts to address some of the key challenges faced by the research and commercial communities such as:

1. Improve Researcher/Developer productivity
2. Reproducibility of results
3. Collaboration between research/development teams
4. Portability and interoperability between different computing environments and scientific domains

Simulation postprocessing utilities are an example suite of applications that workflow systems can add value to. These utilities generally read output file(s) produced by a numerical solver during a simulation, process and plot data. They play a crucial role in assisting the user to visualize, analyze and interpret the simulation outputs.

A simulation can produce numerous output files, and they may be text-based or binary. The format and data captured in each file may be different from one another. Simulation outputs typically include time-series data of various state variables such as pressure, velocity, density, etc. as well as simulation performance and monitoring data. These indicate how the system and the simulation of the system changes over its duration. A typical utility will make use of a combination of specialized operations that together:

1. Reads one or more of these outputs.
2. Processes the data as required.
3. Optionally “smooth” the data if required.
4. Plots the necessary data as charts for visualizing.

While all utilities perform the above steps, what is involved in performing each step might be very different because:

1. The reader operation may be different because the output data being read is different
2. The data processing steps may be different because the reader outputs in a different format
3. The code to smooth the data and to plot the data may be different owing to the data format, and the type of smoothing required.

This means that the smoothing of data and plotting of data will essentially be unique to each utility and will likely be largely self-contained within the utility. But the challenge from an efficient workflow development perspective is to develop the new capability (time-series smoothing here) in a generic form with high levels of re-use and minimal customization when it is embedded into applications within the broader utility software family.

2. A GENERIC TIME-SERIES SMOOTHING OPERATION

Smoothing of the output data may be required at times to remove the sudden or sharp undulations in the plot of a scalar variable. These undulations may be perceived as "noise" and smoothing these out gives a better (more interpretable) plot that helps the user to better analyze the actual signal better. Two types of smoothing algorithms (Brownlee et. al. (2020)) are generally used by the post processing utilities:

1. Exponential Smoothing, and
2. Overall Mean Smoothing

Analysis capabilities for research modelling and simulation are continually evolving over time. As new requirements accumulate over time, the cost of updating existing analysis software tools may become too high. If the cost of updating these tools is high, then this will constrain the rate of development of that modelling area and limit the ability of users to optimally interact with and analyze their data.

In our case, as a large number of utilities were evolving, the need to have a generic "Time-Series Smoothing" operation that could smooth time-series data using different algorithms across a range of utilities became critical. An important requirement was that the operation could be applied to time-series data in any workflow to produce a "smoothed" version of the raw data with minimal time and effort. Existing workflows that belong to the various utilities could then use this generic operation in lieu of the prior specifically customized smoothing code.

To address this need, we have created a generic smoothing function inside a utility that works specifically with the specialized data types used by the utility. As more utilities need their own smoothing function, each devise their own version of the smoothing and each work with different data types (e.g. vector, list, ObjectArray, ObjectDictionary etc.). As the utilities will evolve and new ones will be added, the requirement for a more generic time-series smoothing operation has arisen. This operation needs to work with any data type recognized within the Workspace ecosystem.

This is not as simple as having the smoothing code accept templatized input, because various utilities could use different data types (including data types that will be introduced in the future), and all these types may not be known while the generic time-series smoothing code is written.

In essence:

1. The smoothing operation should work with all data types that exists within the Workspace ecosystem.
2. It must be relatively easy for us make the smoothing operation work with newly introduced data types using various data conversion options and scripting features available within Workspace.
3. It must be relatively easy for us to add the new smoothing capability into an existing Workspace based application.

3. DIFFICULTIES IN REUSING EXISTING CODE

As there are varying forms of output data that can result from a simulation, each utility has its own version of the smoothing code and requires input parameters to tell the smoothing code among other things, what type of smoothing algorithm is to be used and the start and end times to perform smoothing, and so on.

The difficulties involved in attempting to reuse the smoothing algorithm in one of the existing utilities are:

1. Parts of the smoothing code could be reused if the data involved is similar, but this is only possible to a certain extent. As the types of output data formats increase with time, the existing smoothing code will become obsolete.
2. A bug fix or an improvement made to, or a feature added to the smoothing code will need to be manually added/adjusted to the relevant section of each utility, which is laborious and prone to errors and therefore inefficient.
3. Any changes resulting from this smoothing code may also require modifications to other analysis and visualization code which could require significant increase in effort.

4. DIFFICULTIES IN RETROFITTING THE NEW SMOOTHING CODE INTO AN UTILITY

The requirements for modelling and simulation software normally evolve with time due to the changing nature of the problem being simulated or with what needs to be extracted from the simulation. This means further capabilities of various kinds need to be added to existing software. This is especially true when the software is built incrementally, which is common for developments over the long term and for developments with large research components.

The issues we faced in developing a new capability that could be retrofitted into existing software are outlined below:

1. The new code will need to support many data types for the input field that represents the parameters that change over the time series. This is because each utility may deal with one or more types of "readers" that will output very different data types.
2. Building an adapter for the possible types of data might be a time-consuming task and may require further work as soon as newer output formats arise.
3. Attempting to use fixed conversion routines that convert data may also suffer from the same shortcomings.

5. THE SOLUTION - STANDARDIZE INPUTS AND USE OF THE NEW OPERATION

The first step to having a generic time-series smoothing operation will require the operation input data be standardized. As the existing utilities are built using the Workspace architecture, we can readily make use of the various methods to convert between data types to arrive at the desired standard type. In this specific case, it was decided that the generic time-series smoothing operation will accept the following inputs:

1. A QVector<double> to represent the time data
2. An ObjectArray (an array of [array of double]) to represent the various raw data parameters
3. SmoothingType – Type of smoothing algorithm to be used (Overall Mean or Exponential Smoothing)
4. ExpSmoothing – Smoothing parameter used in the Exponential Smoothing algorithm
5. Start and End times – the time range for which smoothing should be performed

Of the above inputs, 1 and 2 are the ones we must highlight here as those are the standardized data formats for the new operation. Data from other formats will need to be converted into one of these standardized formats in a workflow in order to use the new smoothing operation.

We have added the new capability to several post processing utilities. We will illustrate how the workflows were modified to include this capability into two utilities named "Inflow Monitor" and "Outflow Monitor". These utilities measure volumetric inflows and outflows (in units of m^3/s) in fluid flow simulations.

Figure 1 and Figure 2 show a portion of the workflow that is used to build the two different applications into which the newly built smoothing operation has been embedded. Figure 1 shows a reader operation (InflowMonitorDatReader) that produces an ObjectDictionary type output, which is being standardized into a QVector<double> and an ObjectArray. Figure 2 shows a reader operation (OutflowMonitorDatReader) that produces an ObjectArray, which is being standardized into a QVector<double> and an ObjectArray.

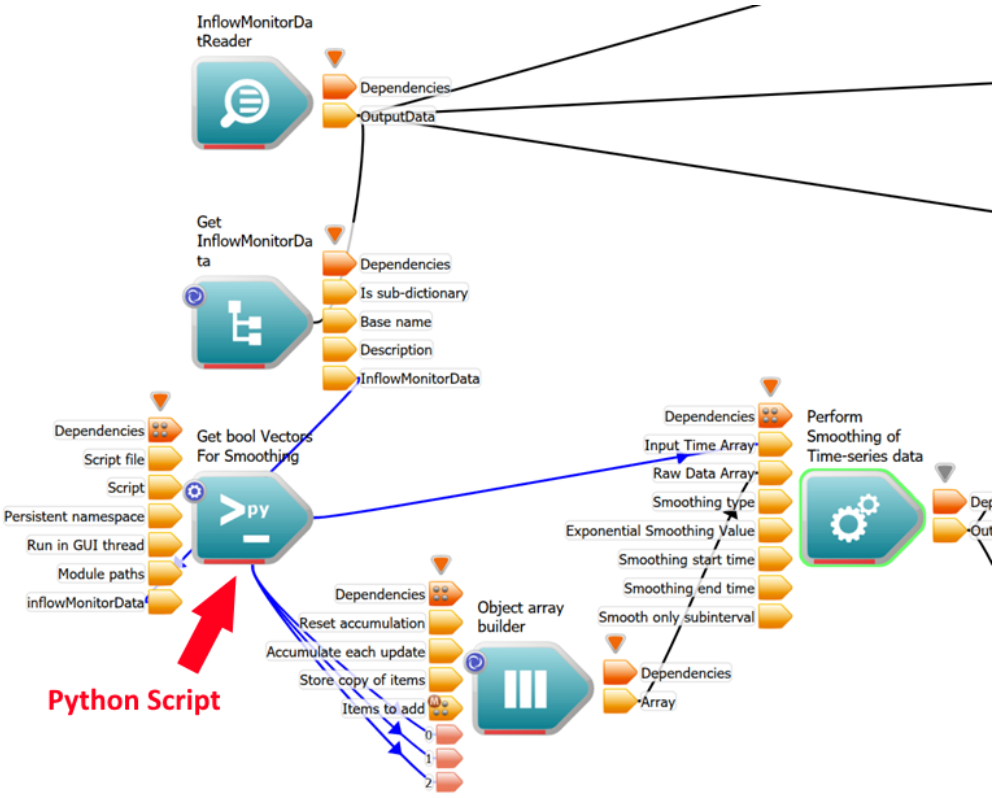

Figure 1. Inflow Monitor Utility - ObjectDictionary being parsed and standardized into a QVector<double> and an ObjectArray**.**

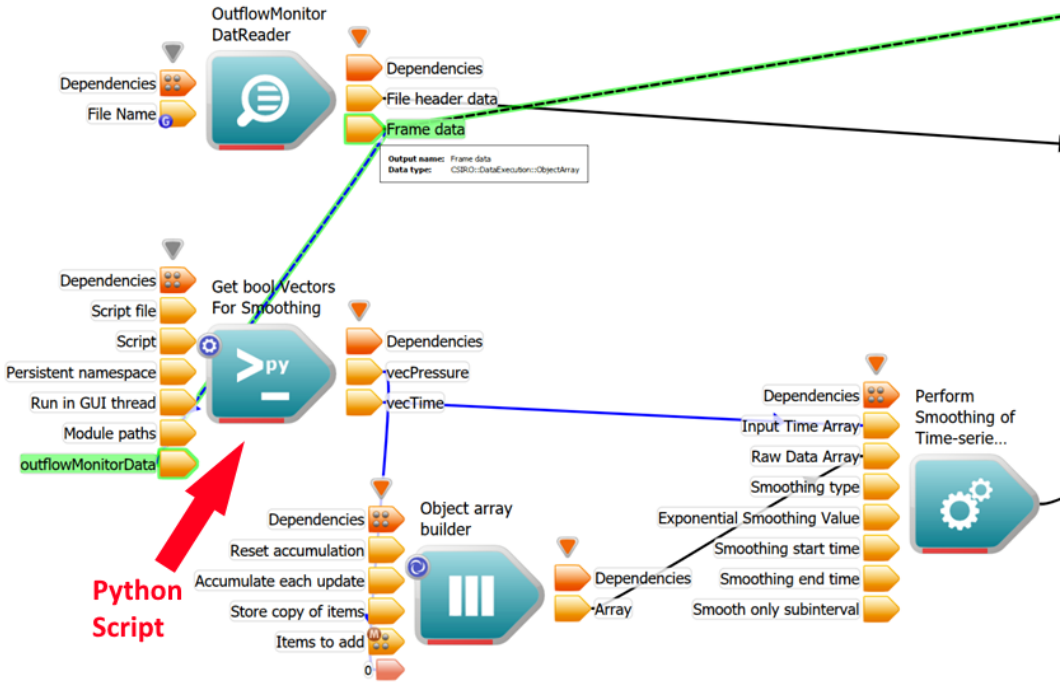

Figure 2. Outflow Monitor Utility - ObjectArray being parsed and standardized into a QVector<double> and an ObjectArray.

Figure 1 and Figure 2 show:

1. The Inflow Monitor utility reads its input file and produces an ObjectDictionary type output.

2. The Outflow Monitor utility reads its input file and produces an ObjectArray type output.
3. In the Inflow Monitor utility in Figure 1, the operation labelled GetInflowMonitorData gets a specific item we want from the ObjectDictionary.
4. We then use a Python Script and an ObjectArrayBuilder operations (both in-built operations) to successfully standardize the data as required by the new time-series smoothing operation.
5. This data is then connected as input to the time-series smoothing operation, which produces the smoothed version of the raw data supplied to it.

Figure 3 shows the workflow of an existing application (Inflow Monitor utility) making use of the newly developed smoothing operation. Inflow velocity, mass and volumetric flux data are now smoothed, and this smoothed data is passed to the pre-existing charting operations for plotting.

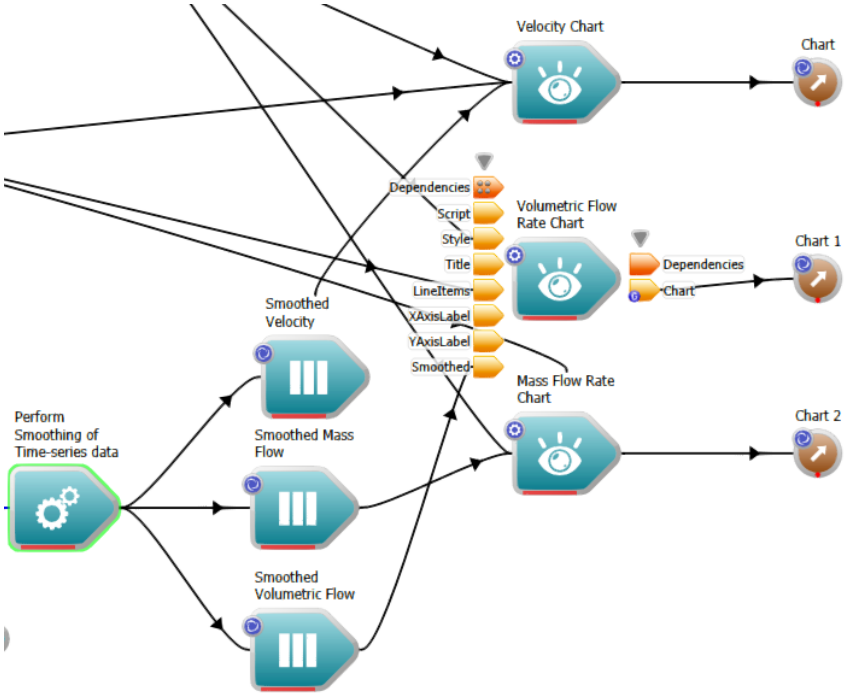

Figure 3. Adding the new smoothing operation into the existing Inflow Monitor utility. Smoothed output data is now produced and plotted with minimal changes to the workflow.

6. UPDATED EXISTING INFLOW MONITOR UTILITY

Figure 4 illustrates the output of the updated Inflow Monitor utility. It now features both raw data as well as the smoothed one in its plot.

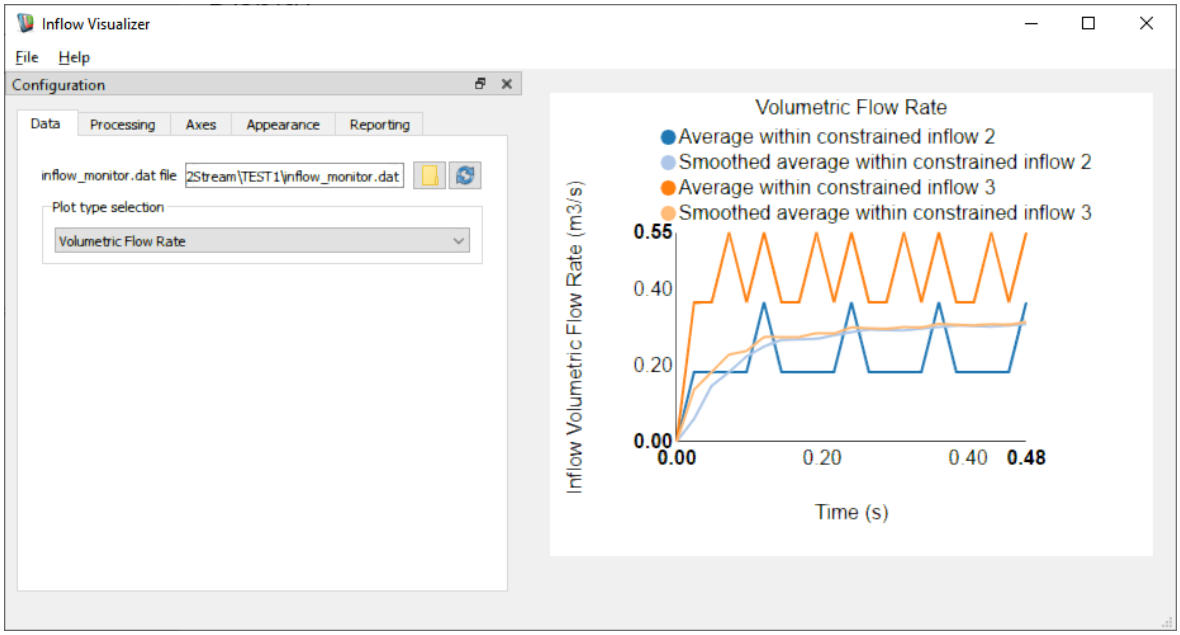

Figure 4. GUI of Inflow Monitor utility. The left panel shows the settings to choose the input file and output data to plot (Volumetric Flow Rate in this case). The right panel shows the plot with raw and smoothed data.

7. CONCLUSION

In this paper we have outlined the process of extending the capabilities of a commercial-grade application using the Workspace workflow platform. This paper illustrates how an existing application built using the Workspace architecture could be easily extended, based on capabilities inherent to the Workspace ecosystem. Benefits of using such a workflow approach were identified as; low financial cost; low resource cost (both in people and in time needed); re-use of existing components from Workspace; re-use and/or customization of existing application-specific components.

REFERENCES

Cleary, P. W., Thomas, D., Hetherton, L., Bolger, M., Hilton, J., and Watkins, D., (2020), Workspace: a workflow platform for supporting development and deployment of modelling and simulation, Published in the Journal of Mathematics and Computers in Simulation, Vol 175, Pages 25-61, Link: http://hdl.handle.net/102.100.100/341873?index=1

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, eResearch Australasia Conference, Melbourne, VIC, Australia, October 2016, Link: http://hdl.handle.net/102.100.100/89557?index=1

Cleary, P. W., Bolger, M., Hetherton, L., Morris, B., Rucinski, C., Thomas, D., and Watkins, D., (2013), Workspace: scientific workflow platform , eResearch Australasia Conference, Brisbane, QLD, Australia, October 2013, Link: https://eresearchau.files.wordpress.com/2013/08/eresau2013_submission_89.pdf

Cohen, C. Z. Raymond., Harrison, Simon M. H., and Cleary, P. W., (2020), Dive Mechanic: Bringing 3D virtual experimentation using biomechanical modelling to elite level diving with the Workspace workflow engine, Mathematics and Computers in Simulation, Volume 175, 2020, Pages 202-217, Link: https://doi.org/10.1016/j.matcom.2020.03.007.

Thomas, D., Anthony B. M., Fiona F. C., Xiang, J., and Feng, Y., (2019), ArcWeld: A case study of the extensibility of software applications built using Workspace architecture, 23rd International Congress on Modelling and Simulation, Canberra, ACT, Australia, 1 to 6 December 2019, Link: https://modsim2019.exordo.com/files/papers/878/final_draft/thomasDG.pdf

Brownlee, J. (2020), A Gentle Introduction to Exponential Smoothing for Time Series Forecasting in Python, Link: https://machinelearningmastery.com/exponential-smoothing-for-time-series-forecasting-in-python