Requirements for Supporting the Iterative Exploration of Scientific Workflow Variants

Lucas A. M. C. Carvalho[1], Bakinam T. Essawy[2], Daniel Garijo[3], Claudia Bauzer Medeiros[1], Yolanda Gil[3]
[1]University of Campinas, Institute of Computing, Campinas, SP, Brazil
[2]University of Virginia, Department of Civil and Environmental Engineering, Charlottesville, VA, U.S.A
[3]University of Southern California, Information Sciences Institute, Marina del Rey, CA, U.S.A

lucas.carvalho@ic.unicamp.br, bte2rn@virginia.edu, dgarijo@isi.edu, cmbm@ic.unicamp.br, gil@isi.edu

ABSTRACT

Workflow systems support scientists in capturing computational experiments and managing their execution. However, such systems are not designed to help scientists create and track the many related workflows that they build as *variants*, trying different software implementations and distinct ways to process data and deciding what to do next by looking at previous workflow results. An initial workflow will be changed to create many new variants thereof that differ from each other in one or more steps. Our goal is to support scientists in the iterative design of computational experiments by assisting them in the creation and management of workflow variants. In this paper, we present several use cases for creating workflow variants in hydrology, from which we specify requirements for workflow variants. We also discuss major research directions to address these requirements.

CCS CONCEPTS

• Information systems → Artificial intelligence; Knowledge representation and reasoning

KEYWORDS

Scientific workflows, workflow variants, computational experiments

1 INTRODUCTION

Scientific workflow systems play a major role in supporting scientists to design, document and execute their computational experiments, automatically tracking provenance during the workflow execution [11; 1]. Scientists follow an iterative exploratory cycle where they often create an initial workflow, and then explore variations of it using different data, replacing some of the software steps, or adding new steps. Sometimes workflows have to be modified because of changes in data (e.g. when datasets are updated with new formats) or software (e.g., software is no longer available, a newer version is better).

In current workflow systems, scientists manage this exploratory process manually. Updating a workflow is a complex and time-consuming task that may involve several steps, and may require tracking down information about different versions of the software used in the workflow.

2017 Workshop on Capturing Scientific Knowledge (SciKnow), held in conjunction with the ACM International Conference on Knowledge Capture (K-CAP), December 4, 2017, Austin, TX.

This paper presents use cases and their requirements to support scientists in the process of exploring different variations of an original workflow, and introduces research directions to address these requirements. These scenarios are based on discussion with domain scientists, particularly in hydrology and bioinformatics.

2 WORKFLOW VARIANTS

Computational workflows describe the computational steps and the dataflow among them to perform complex multi-step analyses. The steps are implemented by *software components* (or *workflow components*) that process data. A software component has a well-defined *interface* consisting of input and output files as well as parameter constant values. The dataflow between components is captured as connections among their respective interfaces. A workflow component may be implemented by a scientist, for example a routine to check for erroneous sensor readings. A workflow component may also be implemented using third-party software, for example invoking a linear regression function from a machine learning software package. A workflow component can be updated in two ways. In some cases, a new *upgrade* of the component is created to override a previous one, for example in cases where the underlying software was corrected to fix a bug. In other cases, a new *variant* of the component is created with new inputs or outputs or other modifications, where the previous versions are still valid and available to the user to use in workflows.

Workflow executions are the result of running workflows and provide provenance for the newly generated data products.

After running a certain workflow, a scientist may want to explore a *workflow variant* that represents a variation of an existing workflow that was run earlier where one or more steps are changed. That step change may require changing other steps that may be affected. In other cases, the scientist may create a new *workflow upgrade* of a previously run workflow that simply replaces a component by a new one with a bug fix. When a workflow is upgraded, the scientist may need to redo previous runs.

Due to the exploratory nature of science, a scientist may start with an initial workflow and iteratively create many workflow variants. During this process, the scientist will want to consider different designs of variants, compare any given variant with previous ones, and synthesize the results of several variants with comparable settings. This iterative process of creating and managing workflow variants is currently not well supported. There are several reasons why a scientist may create a workflow variant:

1. New versions of the software used in the workflow components are released. These may add new functionality that could be useful for the investigation. These may also correct errors or fix bugs, and in that case the scientist may be interested to check whether their results done with the older version still hold.
2. New possible models or algorithms become available. The scientist may discovery these through online search, reading articles, or talking with colleagues. These open new possibilities for exploring alternative designs of the workflow.
3. New datasets become available to the scientist. In this case, the scientist may want to change their workflow to incorporate that new kind of data.

Sometimes the explorations are due to a combination of these. For example, new software versions may fix errors and offer new functionality that allows the scientist to use new kinds of data.

3 RELATED WORK

There have been several efforts to keep track and manage workflow updates and versions. VisTrails [2] tracks the evolution of workflows using a change-based provenance model that records information about modifications to workflow components, inputs, outputs and parameters. They compare results of workflow executions using visualizations. However, this approach focuses on capturing changes and comparing workflows, while we are interested in supporting the process of designing, creating, and managing workflow variants.

Koop et al. [8] focuses on the problem of supporting workflow upgrades when the software that implements a component has a new version by suggesting how the change-based provenance actions might be reused to upgrade other similar workflows. The focus is on the mechanics of the upgrades, while our interest is on supporting the iterative exploration and design of new workflows.

Workflow variants are also explored in Experiment Lines [20]. Their focus is on the variation of models or algorithms and software packages. In contrast, our focus is broader in that we support the creation of workflow variants.

4 MOTIVATING SCENARIOS AND REQUIREMENTS

This section describes several scenarios where scientists iteratively create and explore workflow variants. The scenarios use examples from hydrology. A hydrologist uses models, often developed by others, to estimate how much water will flow in an area. We will consider several hydrology models in these scenarios. MODFLOW is the U.S. Geological Survey's three-dimensional (3D) finite-difference groundwater model that has been developed for several years and has many versions and variants. A major version of the core implementation is MODFLOW-2005 [7] which simulates confined, unconfined, or a combination of confined and unconfined groundwater-flow problems. A major variant is MODFLOW-NWT [9] which uses a Newton-Raphson formulation. MODFLOW has many packages that run different types of simulations depending on the input data selected, so it needs to be configured to use the packages needed to process the desired data. This is done using FloPy [12], a Python package to create, run, and post-process MODFLOW-based models. We will also use MIKE-SHE [6], another computational hydrology model that solves for both saturated and unsaturated zones in groundwater.

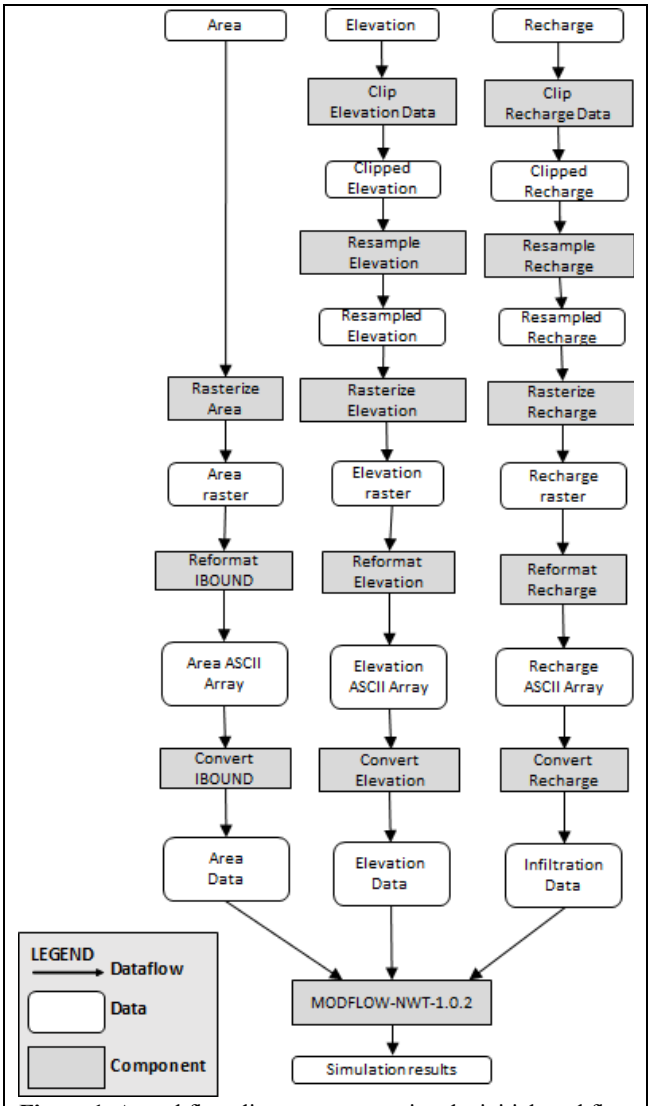

Figure 1. A workflow diagram representing the initial workflow W_0 used in our scenarios. In scenarios S1 and S2 new variants W_1 and W_2 are created by updating the MODFLOW step to use a different version that has the same interface, and therefore has the same overall workflow diagram shown here.

Hydrology models need data about the area for the simulation. For example, MODFLOW requires elevation data, in the U.S. typically coming from the National Elevation Dataset, recharge data, typically from the National Recharge Dataset, and the data for the area from the Watershed Boundary Dataset.

Figure 1 shows an initial workflow W_0 that uses MODFLOW-NWT. The input data includes the boundary for the area being

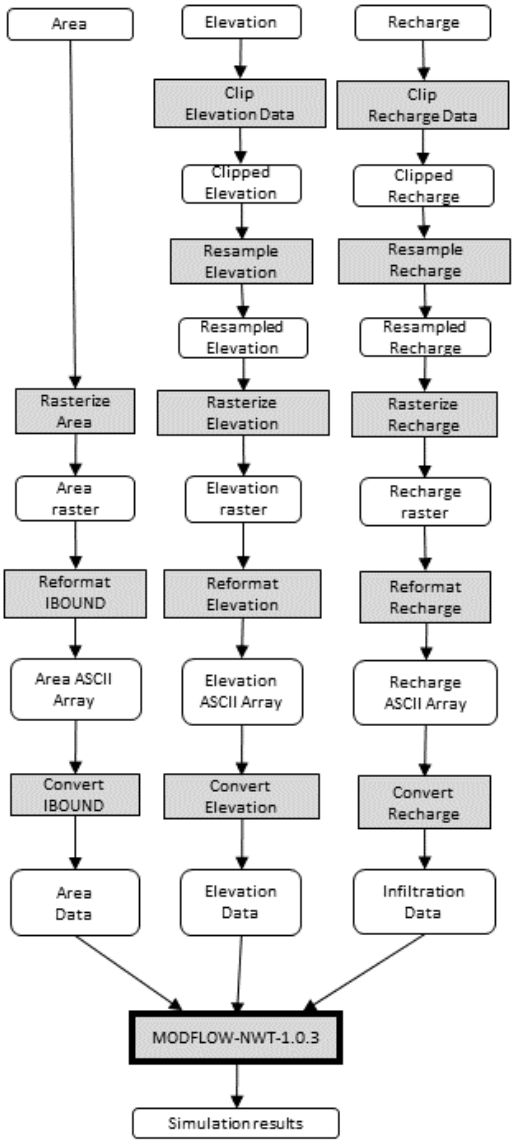

Figure 2. In scenario S1 a new upgrade W_1 is created by updating the MODFLOW component to use a different version that has the same interface and fixes a bug. The changed component is shown with a thicker outline. In scenario S2 a new variant W_2 is created with a similar diagram but using a different MODFLOW component.

studied, elevation data, and recharge (drainage) data. *Area* is an input to the step *Rasterize*, which converts the data from a geographic system format to raster (bitmap) and is implemented using *GDAL* (Geospatial Data Abstraction Library). The step *Convert Area* converts the raster data to text data (ASCII) using *NumPy,* a library for array/matrix in Python. Then, the unit of measurement for this data is converted from centimeters to meters and the data format is converted from raster to text using the step *Convert Elevation*, which is implemented using *NumPy*. *Rasterize Recharge,* which is implemented using *GDAL*, generates the recharge raster and then *Convert Recharge*, which is implemented using *NumPy*, converts the unit of measurement of the data from centimeters to meters and from meter/year to meter/day and from raster format to ASCII. The simulation component is implemented using MODFLOW-NWT Version 1.0.2, and uses FloPy to configure it with the appropriate packages.

4.1 Case I: Same Component Interface, Different Software Version

In this case, a workflow component is replaced by another one that uses a different version of software to implement it but the component interface remains the same. We consider two main scenarios for this case. One occurs when a new version of the software used in a workflow component is released to fix errors or bugs. The other one occurs when a new version is released to carry out a different function.

The first scenario S1 starts with a scientist that runs workflow W_0 several times, changing the data sets used and comparing the results to understand how changes in the inputs influence the results. After several weeks, the scientist notices a new release of MODFLOW, with modifications to enhance the model outputs. So they create an upgrade of the MODFLOW component, which results in the creation of workflow W_1, an upgrade of W_0, shown in Figure 2. In this example, from version 1.0.2 to 1.0.3 of MODFLOW-NWT a bug was fixed in the UZF1 package that was causing UZF1 to incorrectly calculate unsaturated-zone evapotranspiration, which results in a much smaller value [19]. The earlier version 1.0.1 calculated this value properly, so the bug was introduced in version 1.0.2 but fixed in 1.0.3. In some cases, scientists may downgrade to an earlier version because it has a desired feature or it does not produce a wrong value introduced by a bug in later versions but not fixed yet. The scientist may need to discard all previous executions of W_0 because the results were incorrect due to bugs, and run them using W_1 instead.

The second scenario S2 occurs when the software in a component is modified to carry out a different function. In this case, from version 1.0.2 to 1.0.3 of MODFLOW-NWT there is a major change to generate a more accurate calculation of evapotranspiration. First, the header of the listing file that results from running MODFLOW-NWT is changed from having a variable "RMS" to "RMS1" and "RMS2," and from a variable "L2-NORM" to "L2-NEW" and "L2-OLD". This change was done to improve the calculation of the residual terms as the L2-NORM rather than the root-mean-squared error (RMS error). This change does not affect the format of the results, only their value to be more accurate. Thus, the interface of the new component variant does not change, and the newly created workflow variant W_2 has the same structure as W_1 in Figure 2.

The scientist needs to be able to understand the changes to the software in new versions in order to assess the differences between versions and estimate the effort to make the changes in the workflow. This may require a significant effort, as this information may be scattered across release notes, documentation, papers, web sites, and other sources. In some cases, the scientist may be interested in skipping ahead several versions. For example, she

may want to change from version 1.0.2 all the way to the newest version that is 1.1.3. This is challenging since the scientist needs to track and summarize all the differences between several consecutive versions.

In addition, when changing a software version used to implement a component, the scientist needs to check if the new version is compatible with the software version of other components of the same workflow. For example, a specific FloPy version is compatible only with some MODFLOW-NWT versions. Sometimes these incompatibilities can occur across different workflow components, for example if two components make different assumptions about the Newton-Raphson formulation. This means that the scientist needs to track in detail all the software dependencies and compatibilities across the software components of a workflow.

Finally, the scientist may need to check that the new simulation results do not require additional changes in the workflow steps that use those results. In our case they were the output of the workflow, but in other cases further adjustments may be required.

Scenarios 1 and 2 motivate the following requirements:

- **R1 –** Version descriptions need to capture useful metadata of the software.
- **R2 –** Scientists need to understand differences in metadata between different software versions, particularly about their interfaces.
- **R3 –** Scientists need to be alerted about relevant updates of software used in their workflows.
- **R4 –** Workflow descriptions need to capture the software, software version, and functions used in the implementation of workflow components.
- **R5** – Scientists need to understand how new workflow variants can be used to correct errors in prior results.
- **R6 –** Scientists should be able to easily replace a component of a workflow with a new one when the interfaces of the components are the same.
- **R7 –** Given a software package that can be used to create many workflow components, scientists need to easily figure out how to implement new variants of a workflow component with newer versions of that package.
- **R8 –** Scientists should be able to easily create new versions of workflow components and relate them to each other.
- **R9 –** Scientists should be able to easily create new workflow variants and relate them to each other.
- **R10 –** Scientists should be able to relate changes in software to specific workflow results, so it is clear how new software versions affect calculated variables to produce wrong values.
- **R11 –** Version descriptions need to capture bug fixes and known bugs and relate them to software features and input and output file variables.
- **R12 –** Scientists need a summarization of changes between a given software version and a newer version to understand their differences without need to understand the changes associated to each version in between those.

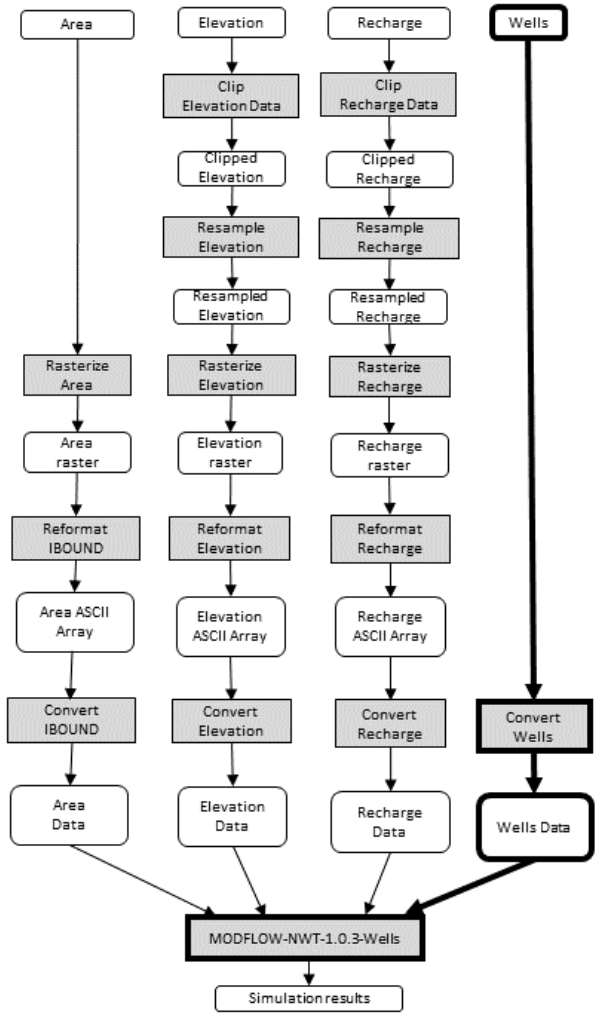

Figure 3. A workflow variant W_3 derived from W_2 is created in Scenario S3 after adding well data, which requires adding the necessary data conversion steps and also changing the MODFLOW component to have an additional input for well data. The new and modified components are shown with a thicker outline.

- **R13 –** Scientists need to understand any incompatibilities between versions of different software packages and libraries used to implement a workflow component.
- **R14** – Scientists need to know whether a new workflow version or a new workflow variant is valid.

4.2 Case II: Different Component Interface, Same Software Version

In this case, a workflow variant is created by replacing a workflow component by another one that uses the same software implementation but invokes a different function and as a result has a new component interface (i.e., adding, removing or replacing inputs or outputs). This interface change may require changes in other steps of the workflow (e.g., adding, replacing or removing

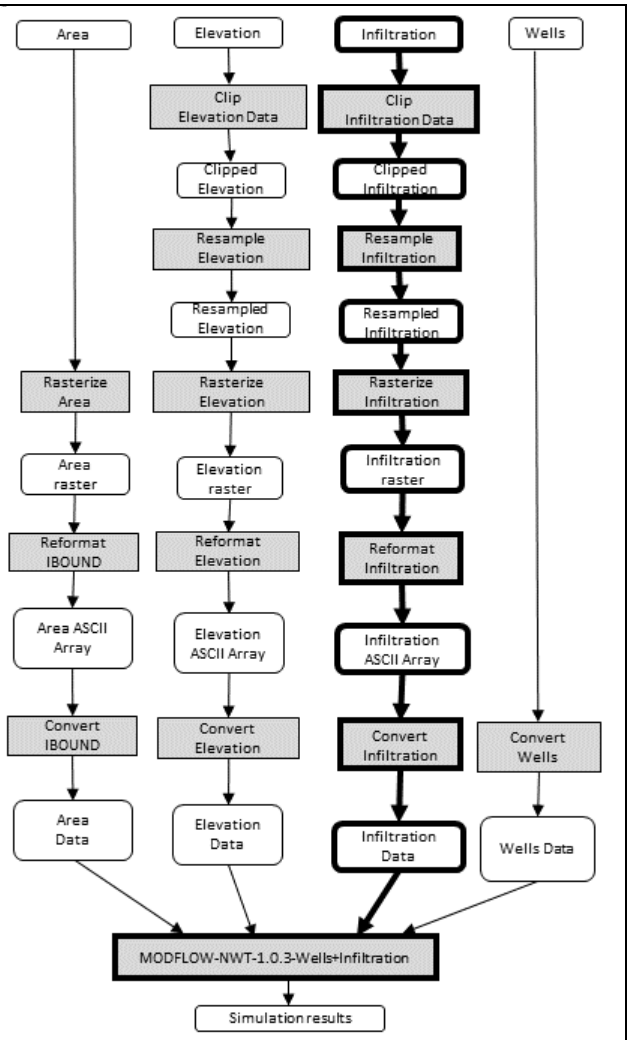

Figure 4. A workflow variant W_4 derived from W_3 is created in Scenario S4 after replacing the input for recharge data with infiltration data, which requires adding the necessary data conversion steps and also creating a new MODFLOW component variant to make the well data compatible. The five new components and the modified one are shown with a thicker outline.

data conversion or post-processing steps.) We consider two scenarios for this case. One occurs when a component is changed to use additional inputs or outputs provided by the software used to implement it. Another occurs when a component is changed to replace inputs or outputs or use them differently in the software used to implement it. In both cases, the rest of the workflow may be affected by the changes.

Scenario S3 starts with a scientist running workflow W_2. The scientist would like to add an input regarding water elevation through wells, so she creates a new variant of the workflow component by adapting the MODFLOW component used in W_2 by adding a new input for well data. The well data is already provided as an ASCII file, so unlike the elevation and recharge data there is no need to convert wells data to ASCII. The only data preparation needed is converting the unit of measurement from feet to meters. To perform this change, the scientist adds *Well* to the workflow inputs, adds the step *Convert Well* for unit conversion. This results in workflow variant W_3, shown in Figure 3. The scientist created one new component variant and created a variant of an existing component.

In scenario S4, the scientist decides to include snowmelt in the simulation. This can be done by using infiltration as an input instead of recharge (since the infiltration package will also account for recharge). Figure 4 shows the resulting workflow variant W_4 where the recharge input of W_3 is replaced with infiltration. Additional changes include replacing the steps to prepare data for simulation with those steps to clip and resample infiltration, and to convert the unit of measurement in the input data from centimeter per year to meters per day, reformatting it to ASCII format. In addition, the MODFLOW step needs to be modified in two ways. First, the recharge input needs to be replaced with infiltration input. Second, the FloPy software configures MODFLOW to use the infiltration packages. In total, the scientist created five new components and created a variant of an existing component.

There are several important tasks that the scientist needs to address in these two scenarios.

Before creating the data preparation components for W_3 and W_4 the scientist has to find whether components that already do those conversions are available or not. Reusing components saves time, but after spending many years running similar workflows with similar data it may be hard to remember which components have been created before. Furthermore, in addition to reusing components it may be possible to reuse entire sub-workflows. In our example, the sub-workflow to prepare infiltration data has five steps that can be reused together.

Another important task is to compare the results of different workflow variants. For example, a scientist would run W_3 and W_4 and compare the results to each other and to W_2 to understand how changes in the workflows affect the simulation results.

Scenarios 3 and 4 motivate these additional requirements:

- **R15** – Scientists need to easily find software packages and workflow components that are appropriate to process a specific type of data input.
- **R16 –** Scientists need to easily find workflow components for data conversion.
- **R17** – Scientists need to be able to understand the differences between two workflow variants.

4.3 Case III: Alternative Component, Different Software

In this case, a workflow variant is created by replacing a workflow component by a component that does an equivalent function but is implemented using a different software. There are several reasons to create workflow variants that use equivalent software, such as testing different models or taking into account parameters that are ignored by the current model used in a workflow. The new component may have a very different interface from the previous one, thus requiring a major update of the workflow to create,

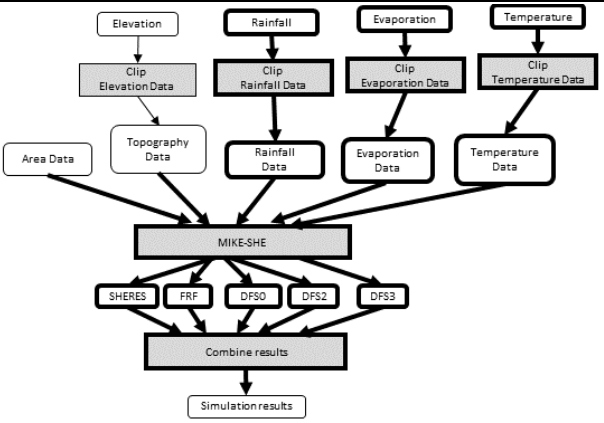

Figure 5. Workflow variant W_5 using the MIKE-SHE hydrological model. The area data can be used raw, and the component to clip elevation data can be reused from workflow W_4. The five new components and the modified one are shown with a thicker outline. Several components from W_4 were removed as they were no longer needed.

replace, or remove several data preparation or post-processing steps. Note that although the tasks to create the workflow variant may be similar to those in scenarios 3 and 4, now there are additional tasks in finding out information about the new software to check its functionality and analyze how it fits into the workflow and the overall exploration that the scientist is doing. We consider two scenarios. One occurs when the scientist already knows which alternative method to use. Another one occurs when the scientist needs to find and compare the assumptions, functionalities and the effort to change the workflow when considering more than one method to decide which one to use.

Scenario S5 has a scientist who is concerned about MODFLOW-NWT only solving for saturated zones, so some parameters are simplified or even ignored. The scientist would like to use a different method to solve for the unsaturated zones, and has heard that the MIKE-SHE model does a similar simulation to MODFLOW, but is a fully coupled and integrated surface water and ground water model that considers parameters regarding unsaturated zones. The inputs and outputs for MIKE-SHE are different from MODFLOW. MIKE-SHE uses area data in the same raw format that is provided by the data source, so it does not need to be pre-processed. It also uses topography data and can ingest formats very similar to the data source, so the data only needs to be clipped. MIKE-SHE also requires several new input data, namely rainfall, evaporation, and temperature, all in the format provided by data sources so they only need to be clipped. MIKE-SHE also generates several separate outputs, including files associated with the simulation (SHERES), a binary output file containing all the static information on the simulation (FRF), and other results stored in a series of DFS0, DFS2 and DFS3 files. As for the MIKE-SHE component, there is no need to use the FloPy software to implement it. Figure 5 shows workflow variant W_5 created from W_4: the scientist had to create three new data pre-processing components, one new component for MIKE-SHE, and one new component to combine the simulation results to obtain a format that is comparable to W_4 and to all the previous workflow variants. Many components from W_4 were discarded as they were no longer needed.

In scenario S6, the scientist decides to investigate other models, since MIKE-SHE is a commercial, proprietary model. There are many other hydrology models available, including PIHM [13], TopoFlow [14], VIC [15], and dozens of others available in repositories such as CSDMS [16]. The scientist starts to investigate which models produce interesting simulation results, and considers how much effort is required to locate the data required by each model, to develop the data pre-processing components needed, and to install and run each of these models. The scientist finds out that PIHM provides Hydroterre [17], a comprehensive data repository that already provides data in the required format, and a PIHM-GIS software to visualize simulation results [18]. The scientist also finds that PIHM requires a solver in order to run, so the simulation component needs to include the solver software in addition to PIHM. The scientist develops a workflow variant W_6 that includes new components implemented using PIHM, Hydroterre, and PIHM-GIS.

It is important to highlight several important tasks done by the scientist in these scenarios. In both scenarios, but particularly in scenario S6, the scientist needs to compare how two models are similar and how they differ in terms of the input data that they use and the output data that they generate. The documentation of models always includes details of the input and output requirements in terms of files and formats. The scientist will want to understand conceptually how the models work in terms of the physical variables used or generated in the model. That is, understanding the inputs and outputs at the file and format level is important, but understanding how model variables map to each of the files is also necessary. This information is usually not included in the software documentation, but in the publications associated with the model. The scientist will need to consult a variety of sources in order to understand how different models compare [10].

Another important task is to understand the assumptions made by the different models. For example, in hydrology some models may assume the Navier-Stokes equations for fluid motion, while others do not. These assumptions are often not captured in the descriptions of workflow components, which focus on the models as software artifacts rather than research artifacts.

In addition, after creating and running the new workflow variants W_5 and W_6, the scientist will want to compare their results to the results obtained with previous workflows W_4, W_3, and earlier ones. This requires that the scientist understands how the model results are related to one another, which requires understanding what modeling variables are generated and included in the simulation outputs.

Scenarios S5 and S6 motivate the following additional requirements:

- **R18** – Version descriptions need to capture assumptions used in software.
- **R19 –** Workflow components, inputs, outputs or parameters in new workflow variants that are no longer needed need to be removed.

Table 1. Summary of requirements from cases.

Category	Requirement	Cases
Workflow component metadata	R1 – Version descriptions need to capture useful metadata of the software.	C1, C2, C3
	R2 – Scientists need to understand differences in metadata between different software versions, particularly about their interfaces.	C1, C3
	R3 – Scientists need to be alerted about relevant updates of software used in their workflows.	C1
	R4 – Workflow descriptions need to capture the software, software version, and functions used in the implementation of workflow components.	C1, C2, C3
	R8 – Scientists should be able to easily create new variants of workflow components and relate them to each other.	C1, C2, C3
	R9 – Scientists should be able to easily create new workflow variants and relate them to each other.	C1, C2, C3
	R10 – Scientists should be able to relate changes in software to specific workflow results, so it is clear how new software versions affect calculated variables to produce wrong values.	C1
	R11 – Version descriptions need to capture bug fixes and known bugs and relate them to software features and input and output file variables.	C1
	R12 – Scientists need a summarization of changes between a given software version and a newer version to understand their differences without need to understand the changes associated to each version in between those.	C1
	R13 – Scientists need to understand any incompatibilities between versions of different software packages and libraries used to implement a workflow component.	C1, C2, C3
	R18 – Version descriptions need to capture assumptions used in software.	C1, C2, C3
Workflow updates	R6 – Scientists should be able to easily replace a component of the workflow with a new one when the interfaces of the components are the same.	C1
	R7 – Given a software package that can be used to create many workflow components, scientists need to easily figure out how to implement new variants of a workflow component with newer versions of that package.	C1
	R14 – Scientists need to know whether a new workflow version or a new workflow variant is valid.	C2, C3
	R15 – Scientists need to easily find software packages and workflow components that are appropriate to process a specific type of data input.	C1, C2, C3
	R16 – Scientists need to easily find workflow components for data conversion.	C2, C3
	R19 – Workflow components, inputs, outputs or parameters in new workflow variants that are no longer needed need to be removed.	C3
	R20 – Scientists need to assess and compare the effort in creating new workflow variants that represent a significant departure from previous ones.	C3
	R21 – Scientists need to find and compare equivalent computational models, including their inputs, outputs, model variables, data formats, and assumptions	C3
Workflow Comparisons	R5 – Scientists need to understand how new workflow variants can be used to correct errors in prior results.	C1, C2, C3
	R17 – Scientists need to be able to understand the differences between two workflow variants.	C1, C2, C3

- **R20 –** Scientists need to assess and compare the effort in creating new workflow variants that represent a significant departure from previous ones.
- **R21 –** Scientists need to find and compare equivalent computational models, including their inputs, outputs, model variables, data formats, and assumptions.

4.4 Requirements summary

The requirements of the previous scenarios can be grouped into three main categories:

- **Workflow component metadata**, which tackles the representation and metadata of workflow components regarding their interface, functionalities and assumptions, and implementation using software packages and libraries. This metadata would also represent the characteristics of the different versions of the software and the different versions and variations of a given workflow component.
- **Workflow updates**, which address the creation of new workflow variants by replacing, adding, or removing workflow components, the propagation of the effects of those changes throughout the structure of the workflow, and the validation of the new workflow variants.
- **Workflow comparisons**, which address the comparison between different software versions, software packages, workflow variants and workflow runs.

Table 1 summarizes the requirements introduced in this section, pointing out the broad categories they belong to and the cases where they occur. Although we adopt the hydrology domain in our scenarios to illustrate the requirements, our requirements are domain-independent. Workflows in any domain have pre-

processing steps, post-processing steps, and major analytic steps [3]. In the case of hydrology, the analytic steps are done using different hydrology models. Other sciences use algorithms rather than models. For example, different clustering algorithms or sequence alignment algorithms would be used in genomics. The requirements outlined here are generally applicable to other domains.

5 DISCUSSION AND FUTURE RESEARCH

Given the state of the art and the requirements from the scenarios, we outline possible research directions for future work:

1. *Describing workflow components and their underlying software.* This includes the creation and adaptation of existing ontologies to capture information about software versions and variants, including software interfaces and features. OntoSoft [5] is an ontology that might be extended to capture relevant information about software versions and variants. It is also important to integrate these ontologies with workflow systems to describe workflow components. Another area of work is to use them to support the creation of workflow variants.
2. *Managing and tracking workflow variants and their differences.* This includes how to compare workflow components and workflow variants regarding their interfaces and functions, and present these results in a useful way for scientists to understand their differences and the implications on experiment results. A possible approach is using multi-media narratives that combine text, graphics, and visualizations to explain the similarities and differences between software versions, software variants, workflow versions, or functions/methods. More importantly, these narratives should be easily customized to the reader's level of expertise and interest. As a starting point our approach may be based in an approach for data narrative generation [4]. Another research area is to manage histories of creation and evolution of workflow variants, and doing so across many users that may benefit from reusing segments or traversals across users.
3. *Designing an interactive framework to support scientists in the exploration and experimentation process through workflow variants.* This includes how to leverage workflow reuse and composition to support the creation of workflow variants. For example, given a new component that needs to replace an existing one in a workflow, suggest what other components may need to be added or removed from the workflow. Other research would involve mechanisms to identify critical and non-critical components in workflows. The critical and non-critical components could be associated to abstractions defined as motifs [3].

6 Conclusions

This paper discusses the need to support scientists in exploring different experiment designs over time. We presented several scenarios where an initial workflow is modified to create workflow variants by replacing, adding or removing workflow steps. We describe the requirements of these scenarios, and grouped them into three categories: workflow component metadata, workflow updates, and workflow comparisons. We also discussed major research directions to address those requirements, including improved frameworks for describing workflow components and the associated software, for managing and tracking workflow variants, and supporting scientists in the iterative exploration and experimentation process through workflow variants.

Acknowledgments. This work was supported in part by a grant from the US National Science Foundation under award ICER-1440323 and ICER-1632211 (EarthCube RCN IS-GEO), and in part by the Sao Paulo Research Foundation (FAPESP) under grants 2017/03570-3, 2014/23861-4 and 2013/08293-7.

REFERENCES

[1] Altintas, I., Barney, O., and Jaeger-Frank, E. (2006). Provenance collection support in the Kepler scientific workflow system. International Provenance and Annotation Workshop (IPAW), pages 118–132. Springer.

[2] Freire, J., Silva, C. T., Callahan, S. P., Santos, E., Scheidegger, C. E., and Vo, H. T. (2006). Managing rapidly-evolving scientific workflows. In Provenance and Annotation of Data, pages 10–18. Springer.

[3] Garijo, D., Alper, P., Belhajjame, K., Corcho, O., Gil, Y., & Goble, C. (2014). Common motifs in scientific workflows: An empirical analysis. Future Generation Computer Systems, 36, 338-351.

[4] Gil, Y. and Garijo, D. (2017). Towards Automating Data Narratives. In Proceedings of the Twenty-Second ACM International Conference on Intelligent User Interfaces (IUI-17), Limassol, Cyprus.

[5] Gil, Y., Ratnakar, V., & Garijo, D. (2015). OntoSoft: Capturing scientific software metadata. Proceedings of the 8th International Conference on Knowledge Capture (K-CAP), 2015.

[6] Graham, D. N., & Butts, M. B. (2005). Flexible, integrated watershed modelling with MIKE-SHE. Watershed models, 849336090, 245-272.

[7] Harbaugh, A. W. MODFLOW-2005, the US Geological Survey modular ground-water model: the ground-water flow process. Reston: US Department of the Interior, US Geological Survey, 2005.

[8] Koop, D., Scheidegger, C. E., Freire, J., & Silva, C. T. (2011). The Provenance of Workflow Upgrades. Third International Provenance and Annotation Workshop (IPAW),Vol. 6378. Springer.

[9] Niswonger, R.G., Panday, Sorab, and Ibaraki, Motomu, 2011, MODFLOW-NWT, A Newton formulation for MODFLOW-2005: U.S. Geological Survey Techniques and Methods 6-A37, 44 p.

[10] Essawy, B. T.; Goodall, J. L.; Xu, H.; and Gil, Y. Evaluation of the OntoSoft Ontology for Describing Legacy Hydrologic Modeling Software. Environmental Modelling & Software, 92. 2017.

[11] Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., Nieva de la Hidalga, A., Balcazar Vargas, M. P., Sufi, S., and Goble, C. (2013). The Taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. Nucleic Acids Research, 41(W1). W557–W561.

[12] Bakker, M., Post, V., Langevin, C.D., Hughes, J.D., White, J.T., Starn, J.J., and Fienen, M.N., 2016, FloPy v3.2.6: U.S. Geological Survey Software Release, 19 March 2017, http://dx.doi.org/10.5066/F7BK19FH.

[13] Qu, Y. and C. J. Duffy. "A semidiscrete finite volume formulation for multiprocess watershed simulation." Water Resources Research 43(8), 2007.

[14] Peckham, S. D. Geomorphometry and spatial hydrologic modelling. In Geomorphometry: Concepts, Software, Applications, Developments in Soil Science, vol. 33, edited by S. D. Peckham, pp. 579–602, Elsevier.

[15] Liang, X., D. P. Lettenmaier, E. F. Wood, and S. J. Burges, 1994: A Simple Hydrologically-Based Model of Land Surface Water and Energy Fluxes for GSMs, J. Geophys. Res., 99(D7), 14,415-14,428.

[16] Community Surface Dynamics Modeling System (CSDMS) Model Repository. Available from http://csdms.colorado.edu/wiki/Model_download_portal.

[17] HydroTerre Data Services. Available from http://www.hydroterre.psu.edu.

[18] The Pennsylvania Integrated Hydrology Model GIS Interface (PIHMgis), http://www.pihm.psu.edu/pihmgis_home.html

[19] USGS. MODFLOW-NWT Release Notes. https://water.usgs.gov/ogw/modflow-nwt/Release.txt

[20] Marinho, A., de Oliveira, D., Ogasawara, E., Silva, V., Ocaña, K., Murta, L., Braganholo, V. and Mattoso, M., 2017. Deriving scientific workflows from algebraic experiment lines: A practical approach. Future Generation Computer Systems, 68, pp.111-127.