International Environmental Modelling and Software Society (iEMSs)
7th Intl. Congress on Env. Modelling and Software, San Diego, CA, USA,
Daniel P. Ames, Nigel W.T. Quinn and Andrea E. Rizzoli (Eds.)
http://www.iemss.org/society/index.php/iemss-2014-proceedings

Intelligent Workflow Systems and Provenance-Aware Software

Yolanda Gil

Information Sciences Institute and Department of Computer Science,
University of Southern California, 4676 Admiralty Way, Marina del Rey, CA 90292,
gil@isi.edu

Abstract: Workflows are increasingly used in science to manage complex computations and data processing at large scale. Intelligent workflow systems provide assistance in setting up parameters and data, validating workflows created by users, and automating the generation of workflows from high-level user guidance. These systems use semantic workflows that extend workflow representations with semantic constraints that express characteristics of the data and analytic models. Reasoning algorithms propagate these semantic constraints throughout the workflow structure, select executable components for underspecified steps, and suggest parameter values. Semantic workflows also enhance provenance records with abstract steps that reflect the overall data analysis method rather than just execution traces. The benefits of semantic workflows include: 1) improving the efficiency of scientists, 2) allowing inspectability and reproducibility, and 3) disseminating expertise to new researchers. Intelligent workflow systems are an instance of provenance-aware software, since they both use and generate provenance and metadata as the data is being processed. Provenance-aware software enhances scientific analysis by propagating upstream metadata and provenance to new data products. Through the use of provenance standards, such as the recent W3C PROV recommendation for provenance on the Web, provenance-aware software can significantly enhance scientific data analysis, publication, and reuse. New capabilities are enabled when provenance is brought to the forefront in the design of software systems for science.

Keywords: workflows; provenance; semantic workflows; intelligent workflow systems.

1. INTRODUCTION

Modern data science requires managing the increasing complexity of data analysis. Scientific workflows are being used in many areas of science to address that complexity [Gil 2009; Gil et al 2007a; Taylor et al 2007]. Workflows represent complex applications as a dependency graph of computations linked through control or data flow. Workflow systems manage the execution of these complex computations, record provenance of how results are generated, and allow end users with little or no programming skills to create applications by reusing pre-defined workflows built by others.

This paper describes the benefits of workflows to scientists by giving examples from the Wings workflow system [Gil et al 2011a; Gil et al 2011b] based on our prior work on ecological modeling reported in [Gil et al 2011c]. First, we describe basic concepts in workflows as a programming paradigm centered on code encapsulation and reuse, software composition, and abstraction. Second, we introduce *semantic workflows* as a new approach that captures semantics and metadata about both data and software. We also introduce *intelligent workflow systems*, such as Wings, that use semantic workflows to assist users in a variety of ways, including validating workflows, setting up parameters, dynamically selecting models that are appropriate for the data, and automatically generating workflow details. Third, we highlight how workflow systems are *provenance-aware software* in that they generate metadata and provenance of results to facilitate inspectability and reproducibility. Finally, we discuss how workflows can improve software stewardship in science.

Additional details and publications for all the topics discussed here can be found at [Wings 2014].

Table 1. Major Benefits of Scientific Workflows

	Problem addressed	***Workflow Approach***	***Benefit to User***
(1)	Scientists familiar with few (one) **programming** languages	Encapsulation of codes from the command line	**Reuse of code in any language or package, across users and labs**
(2)	Scientists who are **not programmers**	Simple programming paradigm focused on dataflow and software reuse and composition	**Non-programmers can reuse existing workflows or compose new ones out of existing codes**
(3)	Before using a model, data has to be correctly **pre-processed** but this takes a lot of work and is not always well documented	Inclusion of pre-processing steps in workflows in order to capture an end-to-end method	**Scientists unfamiliar with a model can easily experiment with it, no extra effort needed to prepare the data**
(4)	Hard to find an appropriate **visualization** for analysis results	Include visualization steps in workflows	**Scientists can easily understand results from running new models**
(5)	**Quality control** and data cleaning are re-implemented by many, and not always properly	Capture common quality control and data cleaning steps in workflows	**Scientists can easily reuse expert-grade quality control and data cleaning methods**
(6)	Hard to keep track when **continuously running** analyses and exploring new codes	Automatic recording of any executions, including codes, parameter settings, and data products	**Comprehensive record of every test done, variants explored, and data results ever generated**
(7)	**Reproducing** other work takes a lot of time and papers often do not contain enough details	Automatic recording of provenance for data analysis results	**Anyone can easily inspect the details of the workflow used to get a result, reproduce it and obtain new results**
(8)	Access to **high performance computing** resources takes substantial effort	Automatic mapping of jobs to resources, optimization, execution monitoring	**Processing data at very large scale is possible even if unfamiliar with high performance computing techniques**

2. HOW WORKFLOWS CAN BENEFIT SCIENTISTS

Workflow systems are designed to make scientists more productive and accelerate the pace of research. This section gives an introduction to workflows and discusses major benefits of workflows to scientists, summarized in Table 1. There are many workflow systems with a wide range of features, here we focus on capabilities supported in Wings and that appear in other workflow systems. For further information about workflow systems and capabilities, see [Gil 2009; Taylor et al 2007].

Data science workflows capture complex data analysis methods as multi-step computations. Workflows are essentially data structures that describe those methods declaratively. Workflow systems can then operate on those workflows and do useful tasks that relieve scientists of the burdens associated with managing the applications manually. Each workflow step is a piece of software (code) that is only responsible for a fragment of workflow functionality so that many components need to be connected to execute the overall data analysis task.

Figure 1 shows a simple workflow on the left to carry out a daily metabolism calculation in a river site for a given time period. It includes a night time model and a reaeration model that are used to calculate metabolism estimates during the night and day respectively. The workflow uses as input hydrological data collected by in-situ sensors for that day. The workflow includes pre-processing steps to clean the data, and a step to create several plots of the metabolism estimates. The top right of Figure 1 shows a screenshot of a workflow editor, with several workflows listed and three workflows shown to illustrate different approaches to data analysis. For example, the workflow in the middle does not include the night time model, and the workflow on the right does not consider a separate reaeration model. The bottom right of Figure 1 lists different runs of some of these workflows, and the details of one of the runs linked to the input data, parameter settings, and results.

(1) Workflows Facilitate Reuse by Encapsulating Heterogeneous Codes. In our experience, scientists are often self-taught programmers that learn one or two programming languages, so they are deterred from reusing code that is implemented in a different language that requires installation of libraries or other setup. This creates a barrier for software reuse. For example, a scientist that is familiar with Fortran will not be able to easily use another scientist's code written in Python. In a workflow, each component can be implemented in a different language, as long as the code can be encapsulated to be invoked from a command line. Figure 1 illustrates this as (1), where the night time model can be in Java, the metabolism calculation in R, and the plotting routine in Python. The workflow system takes care of setting up the execution for each so users do not have to, greatly facilitating code reuse across users and across labs.

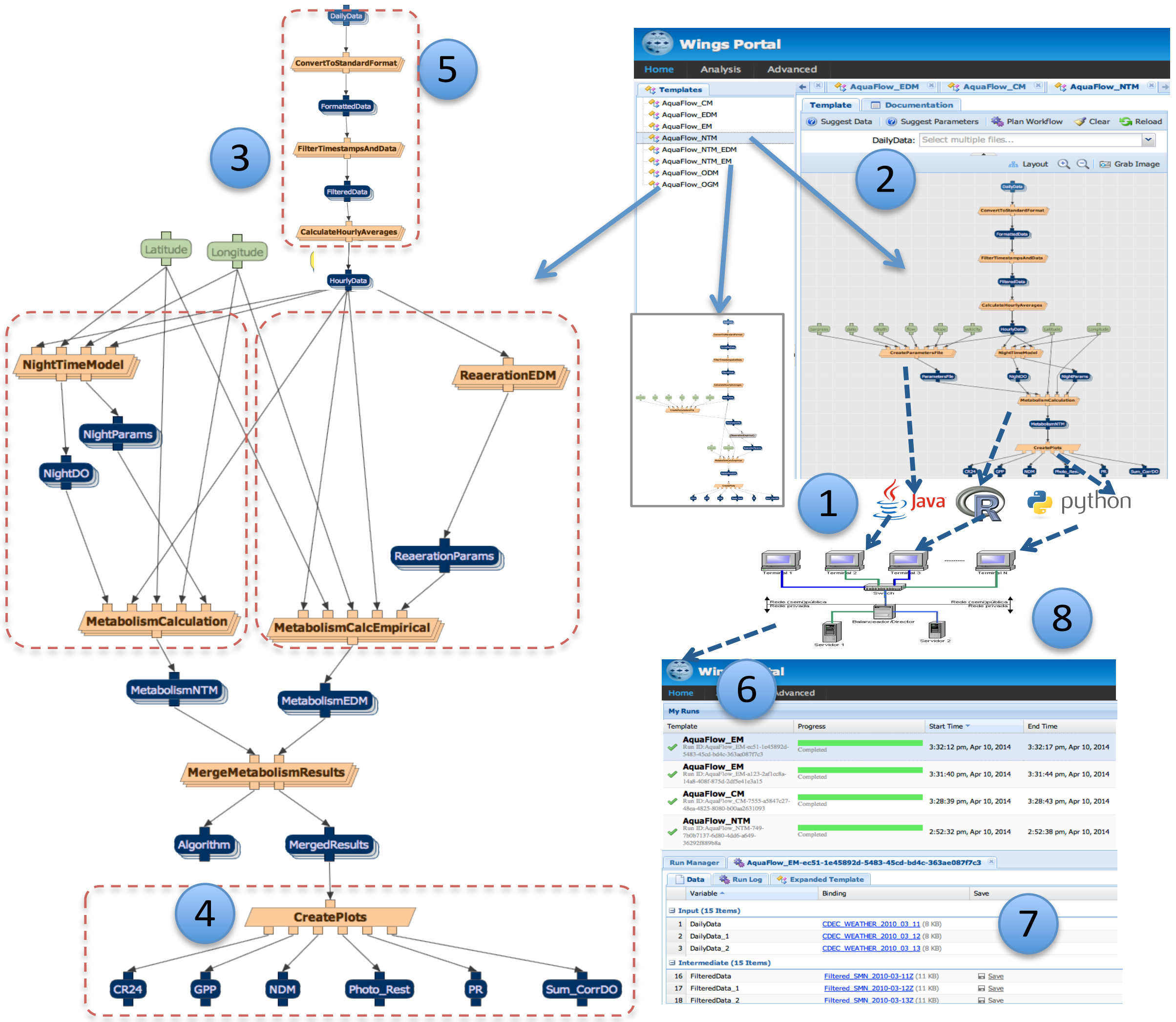


Figure 1. Workflow systems have many benefits to scientists: 1) models and codes implemented in different languages are encapsulated and easily used within the same workflow, facilitating code reuse to others with no expertise in those languages; 2) non-programmers can easily understand and use workflows; 3) workflows can include pre-processing steps necessary to use models, 4) workflows can include steps to generate visualizations of model results, 5) workflows can include quality control and data cleaning steps, 6) workflow runs are tracked by the system to document provenance of results, 7) reproducing prior work is easy through the reuse of a workflow, 8) workflow systems can process large datasets on high-end computing resources (grids, clusters, clouds) behind the scenes.

(2) Workflows Offer a Simple Programming Paradigm that Makes them Accessible to Non-Programmers. Many scientists are not programmers, and yet they would like to run codes to do data analysis. They do a lot of work on spreadsheets, which is very time consuming and prone to error. Although some workflow languages are full-fledged programming languages, some workflow languages are designed to use very simple programming constructs that are intuitive and easily adopted by non-programmers. Wings uses simple dataflow and very limited forms of iteration. Figure 1(2) shows a snapshot of the Wings user interface. We have found that non-sophisticated computer users (at the level of high-school students) can effectively use workflows to solve problems after minimal training.

(3) Workflows Facilitate Data Preparation. Before using a model, the data has to be formatted and prepared to fit what the model needs. Scientists spend considerable amounts of effort just pre-processing and preparing data. It is estimated that scientists spend between 60%-80% of a project's effort collecting and preparing data before doing new science. Our studies of workflow repositories are consistent with these estimates. Workflows can include data preparation steps, as shown in Figure 1(3), which facilitates the adoption of new models. Scientists can use the workflow to try out new models with minimal effort investment.

(4) Workflows Facilitate Data Visualization. Creating appropriate visualizations requires significant effort. Workflows can include data visualization steps, as shown in Figure 1(4). Scientists can use the workflow to run a model and get the visualizations favored by model developers.

(5) Workflows Facilitate Quality Control of Sensor Data. Raw data often contains errors and other problems that need to be corrected. Scientists often do not have the expertise or the time to write adequate quality control and data cleaning codes. Workflows can include well-designed and implemented data preparation steps, as shown in Figure 1(5). Scientists can easily reuse expert-grade methods for quality control captured as workflows.

(6) Workflow Executions Are Automatically Recorded to Document Provenance of New Results. Scientists continuously run analyses, exploring different methods and different parameter configurations and settings. They have to keep track of all these executions manually, often by creating README files or directory structures with dates in them. Workflow systems automatically record the execution of workflows, the data used and generated, the parameter settings, the versions of the codes used, and if desired the intermediate data products. Figure 1(6) shows a screenshot of a browser of execution runs in Wings. Scientists can have a comprehensive history of every test done, variants explored, and results ever generated.

(7) Workflows Facilitate Inspectability and Reproducibility. Reproducing the work that others have done is important to establish baselines and compare improvements with one's own work, understand the details of the method used, and verify the results reported by the authors. In prior work, we proposed an approach to quantify the effort required to reproduce the method in a published paper. In our study, it took three months of work, and according to our colleagues this is not an unusual amount of effort. Studies have shown that many papers lack the information necessary for others to reproduce what was done. Workflows record the workflow used to create new results, which documents their provenance and facilitates inspectability and reproducibility. Figure 1(7) illustrates how the provenance of each result is recorded in detail, so that others can inspect and reproduce the work.

(8) Workflows Facilitate Access to High-End Computing. Instead of investing significant effort in learning to use high-end computing resources (clusters, grids, clouds), users can simply submit their workflow and the workflow system can take care of all the execution details by allocating resources, performing optimizations, and monitoring execution progress.

Benefits of workflow systems also include interfaces with data repositories, access to third party web services, and management of workflow repositories among many others [Taylor et al 2007].

3. INTELLIGENT WORKFLOW SYSTEMS

Intelligent workflow systems use semantic representations to reason about applications and data and offer novel capabilities, summarized in Table 2. Wings is an intelligent workflow system that builds on Semantic Web standards to represent characteristics of workflows, software, and data.

Figure 2 shows a semantic workflow in Wings. It is a simplification of the workflow in Figure 1, as there is no night time model and only reaeration estimates are used to estimate metabolism. Dataset have metadata, shown in yellow. Components have associated constraints as we explain below. Wings includes workflow reasoning algorithms that propagate the constraints within the workflow structure and ensure consistent constraints for any given dataset and component [Gil et al 2011a]. An intelligent user interface uses the results of these algorithms to guide the user [Gil et al 2010b].

(9) Validating Workflow to Ensure Correct Use of Software. Wings has a validation capability, shown in Figure 2(9), that checks the dataflow and all constraints, such as those shown in Figure 3.

Table 2. Intelligent Workflow Systems Offer Additional Benefits to Scientists

	Problem addressed	***Intelligent Capability***	***Benefit to User***
(9)	**Errors** are the norm when programming	Validation of proper composition of components	**System tracks errors so user does not need to do it**
(10)	Models/codes are not always properly used, **documentation** is often hard to follow or remember	Representation of use constraints for models/code	**System is aware of the correct use of models/code and can enforce what designers intend**
(11)	**Metadata** has to be generated by hand for analytic results	Automatic propagation of metadata in workflow	**Analytic results always have associated metadata**
(12)	Model **parameters** are often hard to set up	Some model parameters can be automatically set up by system	**Users do not have to track model parameters, easy to adopt new models/codes**
(13)	Scientists often prefer specific **implementations** when several are available for the same method/model	A workflow step can represent a class of workflow components that is instantiated according to user preferences	**System can ensure the reproducibility of a method even if implemented with different codes**
(14)	Scientists may want to use a workflow in **another platform** or workflow system	Representation of workflow independent of workflow system language and implementation	**Users can reuse the workflow in different workflow systems, and in other kinds of scientific environments**
(15)	Different data requires using **different models** for the same task/analysis	Dynamic model selection based on data characteristics	**Users can process streaming data with a workflow that is automatically and dynamically adapted to the data**
(16)	Need to process **data collections** in parallel but they have different sizes	Workflow language that supports intensional descriptions of sets	**System automatically generates as many jobs as needed to the size of the dataset at runtime**
(17)	Hard to **find data** needed from general data sources	Reasoning about constraints in workflow, derive requirements of input data	**System automatically queries and retrieves input data needed to run a model/code**

(10) Enforcing Correct Use of Models and Code. Models and code are not always properly documented, and even after reading documentation it is hard to remember the details. Intelligent workflow systems can represent the constraints that govern the use of components. For example, a model for reaeration is O'Connors Dobbins, and it requires that the river depth be at least 0.61m. This is a constraint that can be added to that component, shown in Figure 2(10). Wings will check that the data used for the component complies with the constraints.

(11) Automatically Propagating Metadata. Many data catalogs have extensive metadata, but when the data is downloaded and processed to generate new results the metadata for those new results has to be manually added. But this metadata propagation can be automated. In Figure 2(11), the "site" metadata of the input, representing where the sensors were, is the same "site" metadata for the metabolism output. Wings can automatically generate such metadata.

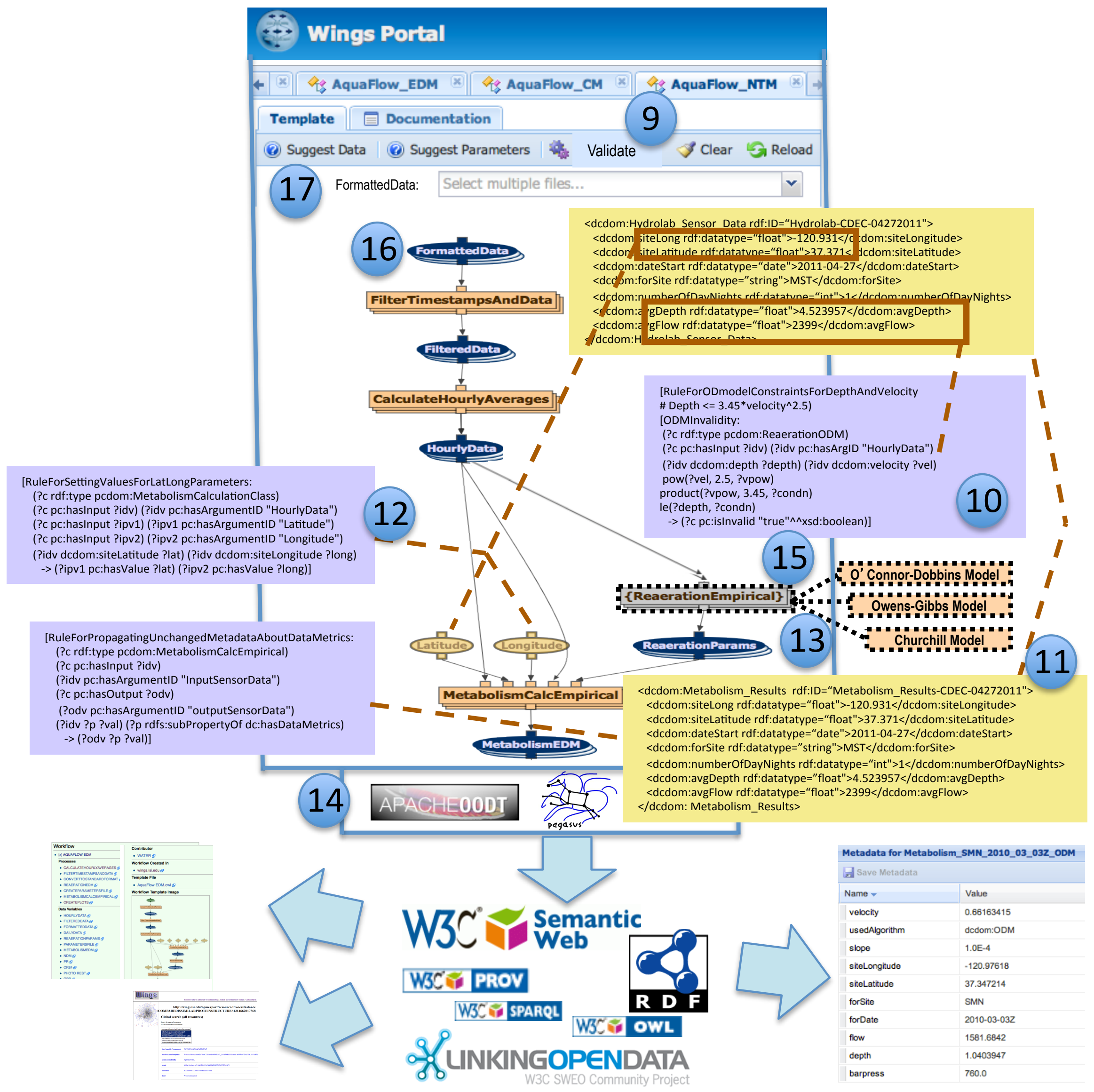
(12) Suggesting Parameter Settings. Some model parameters can be set up automatically given the context of the workflow. In Figure 2(12), the lat/long of the metabolism calculation is the same of the original sensor data. Setting this automatically saves scientists time.

(13) Reproducing Methods Even with Different Implementations. The steps in a semantic workflow can be classes of methods or implementations. When reusing a workflow the steps can be instantiated to the specific running codes available in the execution environment. This facilitates reproducibility, particularly with evolving versions of software.

(14) Reuse of Workflows in Other Platforms. Publishing a semantic workflow as semantic web objects makes it openly accessible and can be imported to other applications.

(15) Dynamic Selection of Models to Suit Data Characteristics. Daily data can have varying characteristics, requiring different models. For example, Figure 2(15) shows three reaeration methods that are each more appropriate depending on the flow, depth, or velocity of the stream. Wings can select an appropriate model based on the daily data.

(16) Processing Data Collections of Varying Sizes. A semantic workflow can include annotations to manage data collections [Gil et al 2009]. Figure 2(16) shows the workflow using with daily data for any number of days, and the system will expand the workflow automatically to parallelize the processing for each day. This saves scientists the effort to write specific scripts for data collections.



```
<dcdom:Hydrolab_Sensor_Data rdf:ID="Hydrolab-CDEC-04272011">
  <dcdom:siteLong rdf:datatype="float">-120.931</dcdom:siteLongitude>
  <dcdom:siteLatitude rdf:datatype="float">37.371</dcdom:siteLatitude>
  <dcdom:dateStart rdf:datatype="date">2011-04-27</dcdom:dateStart>
  <dcdom:forSite rdf:datatype="string">MST</dcdom:forSite>
  <dcdom:numberOfDayNights rdf:datatype="int">1</dcdom:numberOfDayNights>
  <dcdom:avgDepth rdf:datatype="float">4.523957</dcdom:avgDepth>
  <dcdom:avgFlow rdf:datatype="float">2399</dcdom:avgFlow>
</dcdom:Hydrolab_Sensor_Data>
```

```
[RuleForODmodelConstraintsForDepthAndVelocity
# Depth <= 3.45*velocity^2.5)
[ODMInvalidity:
 (?c rdf:type pcdom:ReaerationODM)
 (?c pc:hasInput ?idv) (?idv pc:hasArgID "HourlyData")
 (?idv dcdom:depth ?depth) (?idv dcdom:velocity ?vel)
 pow(?vel, 2.5, ?vpow)
product(?vpow, 3.45, ?condn)
le(?depth, ?condn)
  -> (?c pc:isInvalid "true"^^xsd:boolean)]
```

```
[RuleForSettingValuesForLatLongParameters:
   (?c rdf:type pcdom:MetabolismCalculationClass)
   (?c pc:hasInput ?idv) (?idv pc:hasArgumentID "HourlyData")
   (?c pc:hasInput ?ipv1) (?ipv1 pc:hasArgumentID "Latitude")
   (?c pc:hasInput ?ipv2) (?ipv2 pc:hasArgumentID "Longitude")
   (?idv dcdom:siteLatitude ?lat) (?idv dcdom:siteLongitude ?long)
     -> (?ipv1 pc:hasValue ?lat) (?ipv2 pc:hasValue ?long)]
```

```
[RuleForPropagatingUnchangedMetadataAboutDataMetrics:
   (?c rdf:type pcdom:MetabolismCalcEmpirical)
   (?c pc:hasInput ?idv)
   (?idv pc:hasArgumentID "InputSensorData")
   (?c pc:hasOutput ?odv)
    (?odv pc:hasArgumentID "outputSensorData")
   (?idv ?p ?val) (?p rdfs:subPropertyOf dc:hasDataMetrics)
     -> (?odv ?p ?val)]
```

```
<dcdom:Metabolism_Results  rdf:ID="Metabolism_Results-CDEC-04272011">
  <dcdom:siteLong rdf:datatype="float">-120.931</dcdom:siteLongitude>
  <dcdom:siteLatitude rdf:datatype="float">37.371</dcdom:siteLatitude>
  <dcdom:dateStart rdf:datatype="date">2011-04-27</dcdom:dateStart>
  <dcdom:forSite rdf:datatype="string">MST</dcdom:forSite>
  <dcdom:numberOfDayNights rdf:datatype="int">1</dcdom:numberOfDayNights>
  <dcdom:avgDepth rdf:datatype="float">4.523957</dcdom:avgDepth>
  <dcdom:avgFlow rdf:datatype="float">2399</dcdom:avgFlow>
</dcdom: Metabolism_Results>
```

Metadata for Metabolism_SMN_2010_03_03Z_ODM

Save Metadata

Name	Value
velocity	0.66163415
usedAlgorithm	dcdom:ODM
slope	1.0E-4
siteLongitude	-120.97618
siteLatitude	37.347214
forSite	SMN
forDate	2010-03-03Z
flow	1581.6842
depth	1.0403947
barpress	760.0

Figure 2. Wings is an intelligent workflow system that attaches semantic metadata and constraints to datasets (in yellow) and components (in purple), with significant benefits to scientists: 9) the workflow can be validated, 10) the correct use of models can be enforced based on use constraints, 11) metadata is automatically generated, 12) some model parameters can be automatically set up, 13) the workflow can be reproduced with different implemented codes, 14) the workflow can be used in another platform, 15) the workflow is dynamically configured to suit the data, 16) the system can manage the parallel processing of dataset collections, 17) the system can find auxiliary data as needed for the workflow and the scientist's own data.

(17) Finding Relevant Data. The constraints in a semantic workflow describe the requirements on the input data. Wings can use those constraints to query data repositories, and retrieve ancillary data needed to run the workflow. This saves the scientists time to find and retrieve data.

4. PROVENANCE-AWARE SOFTWARE

A major distinguishing characteristic of workflow systems is that they are *provenance-aware software*. Provenance-aware software generates, maintains, and uses provenance and metadata that describes its output based on provenance and metadata of the input datasets. *Metadata* is associated with data and serves as a description or context for the data. Examples of metadata include use license, format, and statistical properties of data. *Provenance* is a kind of metadata that describes how the data was created. For example the metabolism results in Figure 2 could have the workflow itself as provenance, as the workflow describes the computations that were carried out to generate the results.

Provenance-aware software has, at a minimum, the ability to generate:

1. *A provenance trace* of all the software components used to analyze the initial data. This allows anyone trying to understand how the outputs were generated by inspecting the provenance trace to see the software used, its configuration, and all the intermediate data.
2. *Metadata assertions about new data products* generated. For example, a format converter component that takes an input dataset with location metadata specifying a certain river segment will generate a dataset that should have as metadata also that river segment.

4.1 Use and Generation of Provenance

Although many workflow systems generate provenance traces [Moreau et al 2010], only Wings generates additional provenance that facilitates reuse and reproducibility. Several novel features of Wings support the generation of implementation-independent provenance:

- **Specification of abstract classes of components**. In Figure 2, the reaeration step represents a class of reaeration models, and one is selected at runtime by the system.
- **Specification of types of data** that abstract from particular formats. The datasets in the workflow can be conceptually described based on their contents, rather than their formats.
- **Separation of algorithms from implementation of components**. The steps of the workflow can be component classes, and a particular implementation is selected at runtime.
- **Publication of workflow template** in addition to workflow run. In addition to the provenance record of a workflow execution, the generic reusable workflow can also be exported.

All workflow executions with their provenance can be exported as Linked Data using the W3C PROV standard [Moreau et al 13]. Unlike other provenance systems, Wings provenance records include semantic metadata associated to datasets, which can be used to retrieve workflows based on results.

4.2 Use and Generation of Metadata

In Wings, each workflow component must be described in the workflow system so that it can properly use and generate metadata. Workflow systems always have a description for each workflow component of its input data, input parameters, and outputs. In many cases the types of the data are indicated. In Wings, we have developed several approaches to use and generate metadata. To be provenance aware, each workflow component needs to include constraints in terms of:

- **Use the metadata of its input data to validate that the component is used properly**. For each component, we require a specification of the constraints that it imposes on its inputs. This means that the system has to look at the metadata of the input data and check that it complies with those constraints. For example, for a night time model a constraint could express that the input data must be between dusk and dawn. Input datasets have to be consistent among themselves and with parameters. For example, all datasets have to be for the same location. Any dataset that does not comply with the constraints is not to be used with the component.
- **Generate metadata of its output datasets.** For each component, we require a specification of constraints for how the metadata of its outputs relates to the metadata of its inputs. For example, if the input data was collected for a location, the output metabolism is also for that same location.

One important limitation of Wings that is an active research area is describing the internal structure of data files. In Wings, a file can have associated metadata to describe what it contains. For example, the metadata about the hourly data in our workflow could state that the file contains velocity, depth, temperature, and barometric pressure. However, even if it is a csv file and the data is a structured time series, we cannot indicate which column corresponds to velocity, depth, and the others. Addressing this will require more elaborate representations of metadata.

Wings also does not enforce or encourage the adoption of metadata standards. It would be beneficial to describe software and datasets in terms of standard representations of the domain.

5. IMPROVING SOFTWARE STEWARDSHIP IN SCIENCE

Geosciences software embodies important scientific knowledge that should be explicitly captured, curated, managed, and disseminated. Recurring issues of provenance and uncertainty in the context of data could be better addressed with improved treatment of geoscience software: it is easy to see that nothing describes data more precisely than the software that generates or uses the data. Scientists recognize the value of sharing software to avoid replicating effort or to reproduce results from others. However, the stewardship of software in geosciences must be greatly improved. First, there is very limited, disconnected, and ad-hoc sharing of geoscience software. Modeling frameworks have dramatically improved model sharing, but focus on models with broad interest and are not positioned to attract the many hundreds or thousands of models developed across geosciences. Second, while model software is valued, there are orders of magnitude more codes devoted to preparing data for input to a model and preparing data that results from a model. While the loss of "dark data" in science is well recognized, we see an analogous problem in the pervasive loss of "dark software". It leads to wasted investments, particularly for younger researchers that are often charged with such tasks, which often go unnoticed. Third, the inaccessibility of software as an explicit science product that, like data, should be shared and reused leads to barriers for those who cannot afford such investments, and to barriers for software experts that could otherwise become involved in supporting software efforts. Finally, the education of future scientists crucially depends on the ability to actively explore problems by experimenting with science-grade data and software. This will not be possible unless science software is captured and properly disseminated.

In order to facilitate the sharing of software, we are developing *semantic software repositories* that capture an explicit and comprehensive representation of how software components work. Currently, and even when shared in repositories, software is implicitly linked to its documentation, datasets it uses or produces, publications, and ultimately scientific theories. Our work builds on: 1) the semantic representations of Wings that support sophisticated provenance-aware software capabilities, 2) semantic representations developed of physical variables handled by the software, 3) runtime requirements and licenses for use, and 4) novel approaches to describe the internal structure of data within a file.

Intelligent workflow systems can use these semantic software repositories to significantly accelerate the tempo of scientific research. They offer significant capabilities to scientists, but they need semantic software repositories in order to provide those capabilities. We are currently investigating social and technical challenges involved in creating semantic software repositories and enable the widespread use of provenance-aware software and intelligent workflow systems in geosciences.

6. ACKNOWLEDGMENTS

This research was supported by the US National Science Foundation with awards ACI-1355475 and ICER-1343800. We thank Tom Harmon and Sandra Villamizar for their work on designing the workflows and providing the codes, Varun Ratnakar and Fabio Silva for the integration of the codes as workflow components, and Pedro Szekely, Shubham Gupta, Maria Muslea, and Craig Knoblock for data integration and metadata capture. We would also like to thank Chris Duffy, Scott Peckham, and Chris Mattmann for many discussions on semantic software repositories and geoscience models.

7. REFERENCES

Gil, Y. "From Data to Knowledge to Discoveries: Scientific Workflows and Artificial Intelligence." (2009). Scientific Programming, Volume 17, Number 3.

Gil, Y.; Gonzalez-Calero, P. A.; Kim, J.; Moody, J.; and Ratnakar, V. "A Semantic Framework for Automatic Generation of Computational Workflows Using Distributed Data and Component Catalogs." Journal of Experimental and Theoretical Artificial Intelligence, 23(4), 2011.

Gil, Y., Ratnakar, V., Kim, J., Gonzalez-Calero, P.A., Groth, P, Moody, J., and Deelman, E. "Wings: Intelligent Workflow-Based Design of Computational Experiments." IEEE Intelligent Systems, 26(1), 2011.

Gil, Y.; Szekely, P.; Villamizar, S.; Harmon, T.; Ratnakar, V.; Gupta, S.; Muslea, M.; Silva, F.; and Knoblock, C. "Mind Your Metadata: Exploiting Semantics for Configuration, Adaptation, and Provenance in Scientific Workflows." Proceedings of the Tenth International Semantic Web Conference (ISWC), 2011.

Taylor, I., Deelman, E., Gannon, D., Shields, M., (Eds). "Workflows for e-Science", Springer Verlag, 2007.

Wings. The Workflow System Website: http://www.wings-worfklows.org. Accessed on April 12, 2014.