Using Workflow Medleys to Streamline Exploratory Tasks

Emanuele Santos[1,2], David Koop[1,2], Huy T. Vo[1,2], Erik W. Anderson[1,2], Juliana Freire[2], and Cláudio Silva[1,2]

[1] Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT, USA
[2] School of Computing, University of Utah, Salt Lake City, UT, USA
{esantos,dakoop,hvo,eranders,juliana,csilva}@cs.utah.edu

Abstract. To analyze and understand the growing wealth of scientific data, complex workflows need to be assembled, often requiring the combination of loosely-coupled resources, specialized libraries, distributed computing infrastructure, and Web services. However, constructing these workflows is a non-trivial task, especially for users who do not have programming expertise. This problem is compounded for exploratory tasks, where the workflows need to be iteratively refined. In this paper, we introduce *workflow medleys*, a new approach for manipulating collections of workflows. We propose a workflow manipulation language that includes operations that are common in exploratory tasks and present a visual interface designed for this language. We briefly discuss how medleys have been applied in two (real) applications.

1 Introduction

The trend towards service-oriented architectures has expanded to a number of domains. Recently, a new class of tools have emerged that help users to leverage and integrate services in a collaborative fashion. Yahoo! Pipes [1] is an example of mashup builder that provides a graphical user interface for assembling pipelines that combine RSS feeds and Web services. Scientific workflow systems, such as Taverna [2] and VisTrails [3], provide a more comprehensive framework which, in addition to services, also supports the integration of general tools and libraries.

The ability to construct complex applications, be they scientific workflows or Web mashups, by weaving services together is very appealing and has many benefits. Although workflow systems are natural candidates as solutions to this problem, there are two important challenges that need to be addressed: usability and support for exploratory tasks. While there has been substantial work on workflow and application integration systems [4], such systems have primarily been designed for power users in enterprise settings. Scientists who use scientific workflow systems do not necessarily have programming expertise. Thus, it is not reasonable to assume that they can write complex control-flow specifications (*e.g.,* using languages such as BPEL [5]), even if a visual programming interface is available.

In addition, workflow systems have been traditionally used to automate (complex) processes, which often require a laborious, time-consuming design cycle. In a number of new applications, however, workflows are assembled for exploratory, and sometimes one-of-a-kind tasks. Instead of designing a single workflow that will be run thousands of times, a user (or set of users) manipulates ensembles of workflows that are iteratively refined as she formulates and test hypotheses [6]. Such tasks may require, for example, experimenting with different combinations of parameter values, data sets, or algorithms. Consider the following example of an exploratory task.

Example 1 (Selecting the best learning classifier) To build an effective learning classifier, a user must often tediously build and compare alternative learning techniques as well as experiment with different configurations for each technique. For text classification, support vector machines (SVMs) can be extremely effective [7], but their accuracy depends on a variety of model parameters including the kernel function f, the scaling factor γ, and the penalty parameter of the error term C. Different kernel types (linear, polynomial, radial basis function (RBF), and sigmoid) need to be investigated and parameters tuned for each. Thus, selecting the best classifier requires constructing several classifiers, testing them all, and comparing their error rates. Figure 1(c) shows the accuracy rates of three distinct SVM classifiers using C values 0.25, 1, 2, 4 over the test data.

Suppose a workflow designer constructs three workflows, one for each kernel type. Figure 1(a) shows the structure of one of these workflows. This workflow constructs a classifier using training data retrieved from the Web and computes its error rate. Using a similar process, it also derives the error rate for the classifier on the test data. The error rates from the two runs are then sent to a Matplotlib [8] module which generates the plots that are subsequently displayed on the screen. Using a visual programming interface (such as the ones provided by workflow systems [2, 9]), to compare the different configurations for C values 0.25, 1, 2, 4, the user has to modify, run and save the results of each workflow. This scenario requires 12 modifications and 12 saved files. Furthermore, if new test (or training) data is made available, the whole process needs to be repeated.

Workflow Medleys. In this paper, we propose a new approach to support exploratory tasks that involve ensembles of workflows, or *workflow medleys*. This approach relies on simplified views of workflows that are more intuitive for users, along with operators for manipulating the workflows as a set.

For the scenario above, we desire to create a simplified interface for the given workflows. Note that even small workflows, like the one shown in Figure 1(a) can have many different modules, connections, and parameters, but for most tasks, only a small subset needs to be manipulated. As illustrated in Figure 1(b), simpler views of these workflows might hide all but this subset of entities. In our example, these might include the names of the input files (`dev_file` and `train_file`) and the C parameter (`cost`). Then, to compare the different classifier configurations, as Figure 1(c) illustrates, we can *synchronize* the appropriate parameters across workflows, and then set all of these parameters to the appropriate values. Note that setting the value of a single parameter, *e.g., cost*, updates all parameters that are synchronized with it. This provides a means to efficiently compute and compare the 12 different methods. Also, if new test data is made available, all plots can be re-generated by updating a the `dev_file` parameter only once.

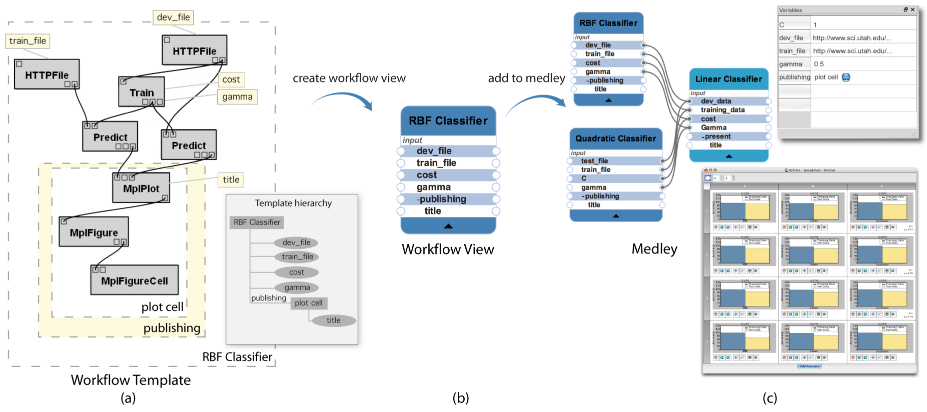

Fig. 1. Example of creating a medley for Example 1. (a) The developer marks configurable pieces of the workflow to create a template. (b) A workflow view is created based on the workflow template. (c) A medley of SVM Classifiers. The connections mean that synchronization is taking place.

Note that the same mechanism used to explore parameter spaces can be used to update the workflow definitions in bulk. For example, by synchronizing the subworkflow (variable `publishing` in the medley shown in Figure 1(c)) which consists of the modules responsible for displaying the results, a single update operation can be used to replace this subworkflow with a different set of modules. For example, instead of displaying the results on the screen, the new version may generate an HTML page with the images embedded.

Outline and Contributions. We propose a new approach that streamlines exploratory tasks that require the composition of multiple workflows. This approach is general and can be combined with existing workflow and workflow-based systems, such as Yahoo! Pipes, Taverna and Kepler. In addition, it can be naturally mapped into an intuitive interface that is suitable for users that are not expert programmers. We introduce the medley model in Section 2. This model consists of a set of concepts and operations for manipulating workflows and captures operations that are common in exploratory tasks. In Section 3 we discuss our first prototype of a user interface for the medley model. We describe how it is implemented and used. We have explored the use of medleys in two real applications: chemical informatics and a comparative analysis of isosurface extraction algorithms. We describe our experiences in Section 4. In Section 5, we review related work and we conclude in Section 6.

2 Manipulating Workflow Specifications

In this section, we show how manipulation of a workflow collection can be simplified by developing the concepts of workflow templates, workflow views, and medleys. We begin by reviewing the definition of a workflow along with basic workflow operations, and then introduce the *workflow template* as a way for designers to designate configurable pieces of a workflows. A *workflow view* is the projection of a workflow according to a workflow template, and a *medley* is a collection of workflow views along with a set of links between them that synchronize or compose the views. Throughout this section, we assume a dataflow model for workflows [10]. Note that all of the operations and concepts we introduce are independent of the underlying workflow management system.

2.1 Workflows

Definition 1. A *workflow* $w(M, C)$ is a set of *modules*, M, along with a set of *connections*, C, linking the modules. Each module $m \in M$ is associated with a tuple (I_m, O_m, P_m), where I_m corresponds to a set of *input ports*, O_m corresponds to a set of *output ports*, and P_m is a list of *parameters*. Each parameter $p \in P_m$ is associated with a value v. A *connection* (o, i) links an output port o from a module m_1 to an input port i of another module m_2. $o \in O_{m_1}$ is the *source port* and $i \in I_{m_2}$ is the *target port*. m_1 and m_2 can only be connected through ports o and i if the *types* of the ports are compatible. *Sources* are modules where no target port is connected, and *sinks* are modules whose no source port is connected. Parameters can also have a type, and the value of a parameter must be an instance of that type. ■

Definition 2. Given a workflow $w(M, C)$, a *subworkflow* $w_s(M', C')$ is a workflow where $M' \subset M$ and $C' \subset C$ such that $c \in C'$ if and only if c connects m_1 to m_2 where $m_1, m_2 \in M'$. ■

While there are a variety of workflow operations we could discuss, we will highlight two: *enactment*, executing a workflow, and *substitution*, changing workflow components. These operations can be used both when designing a workflow and when interacting with a completed workflow.

Enactment. A *workflow enactment* is the execution of a workflow in the order determined by the network of modules and connections. We recursively update modules starting with the sinks until all modules are "up-to-date". Because each module depends on all of its inputs, these data requests propagate all the way to the sources (which reference the initial data), who update their outputs, allowing modules connected to their output ports to then execute. Execution continues until each sink has been executed.

Substitution. *Substitution* allows workflow components (*e.g.,* parameter values and modules) to be replaced. More formally, given a workflow $w(M, C)$, the operation *substituteParameter*$_w(m, p, v)$ assigns value v to a parameter p of a module m in workflow w, provided that the types of v and p are compatible. Given a second workflow $w_s(M', C')$, the operation *substituteWorkflow*$_w(M^*, w_s)$ replaces the subworkflow induced by the modules in $M^* \subset M$ with the workflow w_s. In order to accomplish this, we must create the connections that link w_s back into the workflow w. Each connection that links the modules in M^* to those in $M - M^*$ is remapped to a connection linking M' to those in $M - M^*$ by matching the types of the ports in the original connections to match those in the new connections.

2.2 Simplifying Workflows

As outlined earlier, workflow systems allow users to create and execute workflows. A limitation of these systems is the difficulty involved in modifying an existing workflow by users other than the original workflow developer. Our goal is to simplify these modifications and allow users to interact with ensembles of workflows in a more intuitive manner. Our approach is to allow the designer to designate configurable pieces of the workflow through workflow templates. Such designations help users determine proper inputs as well as experiment with different workflow variations. From such templates, we can create workflow views that abstract much of the complexity of workflows. Users can then combine workflow views in medleys using synchronization and composition operations.

Workflow Templates. We introduce a *workflow template* as a workflow that allows designers to define reconfigurable pieces of the workflow in a hierarchical way. Users can select and label parameters or subworkflows using a nomenclature that is meaningful for a given application or task. Figure 1(a) shows a workflow template generated for the classification workflow described in Example 1.

Note that the designer selected a subset of parameters as well as the plotting subworkflow that should be exposed. The root of the template hierarchy represents the workflow, and its children and descendants correspond to configurable parameters and subworkflows. We refer to each element in the the template hierarchy as a *workflow template node*. Nodes that correspond to subworkflows are represented as rectangles and parameters as ellipses. Note that labels are unique in a given hierarchy level. By representing the template as a hierarchy, our approach is able to handle arbitrary nesting of workflows.

Workflow template nodes provide the same operations of a workflow as well as other specific operations for labeling and removing labels, for creating, adding and removing child nodes, creating and removing connections between template nodes and between template nodes and modules, and for materializing a workflow.

Workflow Views. In a workflow template, important and configurable elements (*i.e.,* modules, parameters, and subworkflows) are selected, and a workflow view effectively hides all unselected elements. More formally, a *workflow view* w_v is a projection of a workflow w where only a subset of the workflow elements are exposed for direct interaction. We refer to the exposed elements as *variables*. Any workflow element not exposed by a workflow view cannot be directly changed in the view. However, a workflow view maintains a reference to the original workflow, and thus views can be enacted by enacting the underlying workflow. Notice that a workflow view can be generated from a workflow template. In fact, the parameters and configurable subworkflows are also represented as a hierarchy that mirrors the one for the template hierarchy. Figure 1(b) shows a view (`RBF Classifier`) derived from the template in Figure 1(a).

Medleys. For exploratory tasks, a user often needs to create and manipulate a set of workflows, as shown in our machine learning example. To support this, we introduce a *medley* $\mathbf{M}$ as a collection of (related) workflow views along with a set of relationships between the views. These relationships are defined by operations linking the views, including synchronization and composition.

When two views are *synchronized*, one or more variables from each view are linked. A variable x in a workflow view $w_v \in \mathbf{M}$ can be synchronized with any variable x' in another view $w'_v \in \mathbf{M}$ if x and x' have the same type. Then, for any pair of linked variables, binding either to a value v ensures that each variable is set to v.

The ability to synchronize variables is useful for tasks like comparative visualization since we can ensure that parameters across different workflows whose values should be the same will indeed be the same. Consider again the machine learning example, and suppose we have a medley with views for the workflows that use the different classifiers. By synchronizing their input files and cost values, a user could quickly set these parameters once and their values would be automatically propagated to the three workflows. Furthermore, synchronization enables a user to efficiently explore different configurations. Instead of setting values for each workflow individually—which can be both time consuming and error prone, the value for a parameter is set only once and is automatically propagated to all synchronized variables in multiple views.

Two views are *composed* by connecting an output port in one view to the input port of the other. In our example, composition could be used to pass the HTML file generated by the two classifier views to a view that sends files to a web server via FTP. In addition, a medley can combine composition and synchronization to easily construct a variety of analyses and explorations.

Note that we could consider synchronization or composition on workflows instead of workflow views, but this could be much more complicated for the user. Because workflow views reduce the number of components that are exposed, they make it much easier to identify how workflows can be integrated and synchronized.

3 Creating and Interacting with Medleys

While workflow templates, workflow views, and medleys allow users to simplify and integrate workflows, constructing these concepts needs to be straightforward. For this reason, we have implemented these operations using an intuitive user interface. In this section, we describe our initial implementation of such an interface.

Creating Workflow Templates and Views. Developers use the *Workflow Template Editor* to create a workflow template, by selecting and labeling parameters and sub-workflows, as shown in Figure 1(a). Given a workflow template, displaying the corresponding workflow view requires a simplified interface. In our implementation, we use a table-based layout where each variable name and editable value are displayed (see Figure 1(c)).

Once a template is created, one of the operations supported by the Template Editor is view creation. While configuring a view, users can set the visibility of the parameters and configurable subworkflows, as well as select suggestions from the list stored in the template. These suggestions will guide the end users to pick meaningful values for the parameters when they are not familiar with the workflows. Note that both templates and views can be stored in a repository where they can be accessed later.

Creating and Manipulating Medleys. To combine workflow views in a medley, the developer uses the *Medley Editor*. The views stored in the Workflow View Repository are displayed on a panel and they can be dragged and dropped on a canvas. Once on the canvas, the medley operations (*i.e.,* synchronization and composition) can be applied to the views. A screenshot of part of the Medley Editor is shown in Figure 1(c).

Each variable in a view has an associated handle (see the circles on the left and right of each variable name in the workflow view in Figure 1(b)). By connecting the handles for two variables in two distinct views, their values are synchronized. To simplify the task of identifying variables to be synchronized, when the developer starts to create a connection all the variables that are compatible with that variable are highlighted.

Demonstration Overview. In this demonstration, we will use this interface to create and manipulate medleys for exploratory tasks in scientific visualization scenarios. In particular, we will demonstrate how to create workflow views and how to synchronize their variables in a medley.

4 Case studies

We tested how medleys can be used in exploratory tasks in two different applications.

Integrating Chemical Informatics Web Services. The first application consisted of integrating chemical informatics web services to locate information about a specific compound and graphically visualize it. To perform this task, a user must invoke several services provided by Chembiogrid [11]. The first workflow fetches the SMILES [3] code of a molecule id. The second and third workflows fetch the 2D image and the 3D model representing the SMILES code, respectively. As the user is not able to render the 3D model in the format returned by the web service, another web service, *sdfToPdb*, is used to convert the data to pdb format. Finally, a fourth workflow is used to render and display the molecule using a ball-and-stick model. Completing this task using a workflow system that supports Web services, such as for example, Taverna, a user needs to assemble a workflow that combines these four workflows, carefully connecting outputs to inputs. In contrast, by creating a medley with workflow views created for the four workflows described above, the user can synchronize and compose the workflows without having to directly modify the structure of workflows.

Comparative Analysis of Isosurface Extraction Algorithms. One of our collaborators needed to perform a comparative analysis of several algorithms for extracting isosurfaces [12], involving the visualization of the meshes produced by the different algorithms and the histograms that accumulate quality information on each mesh. Although a workflow system would help him structure his experiment (*e.g.,* by creating a workflow for each algorithm that both renders the mesh and displays the histogram), it would require him to modify the parameters on each workflow one by one, and repeat this tedious process over and over until a good visualization is found. We created a medley containing workflow views for each algorithm our collaborator wanted to evaluate. He performed the comparative analysis using this medley, by changing parameters and datasets and without having to manipulate the workflows directly. Although the use of a workflow system would help on structuring the experiments, it would still require users to modify the parameters on each workflow one by one, and repeat this tedious process over and over until a good result is found.

5 Related Work

Workflows and workflow-based systems have emerged as an alternative to ad-hoc approaches to data exploration commonly used in the scientific community [9, 2, 13, 3, 14–16]. Workflow systems provide languages with well-defined semantics to specify

[3] SMILES stands for Simplified Molecular Input Line Entry Specification and it is a linear notation that uses alphanumeric characters to encode molecular structure.

computational processes which integrate existing applications according to a set of rules [4, 17–19]. Not only do they support the automation of repetitive tasks, but they can also capture complex analysis processes at various levels of detail and systematically capture provenance information for the derived data products [20]. Workflow systems, however, have been primarily designed for expert programmers and to automate repetitive processes. Our goal with the medley approach is to provide casual users with intuitive interfaces to combine services on-the-fly and perform exploratory tasks through workflows.

Social Web sites and web-based communities (*e.g.,* Flickr [21], Facebook [22], Yahoo! Pipes [1]), which facilitate collaboration and sharing between users, are becoming increasingly popular. An important benefit of these sites is that they enable users to leverage the *wisdom of the crowds*. In the (very) recent past, a new class of Web site has emerged that enables users to upload and collectively analyze many types of data (see *e.g.,* [23, 24]). These are part of a broad phenomenon that has been called "social data analysis" [25]. Many Eyes [23] (developed at IBM research) is a site for sharing and commenting on visualizations. Users can upload any data set and visualize the data using a wide range of tools provided by Many Eyes (*e.g.,* line graphs, stack graphs, bar charts, block histograms, treemaps). This trend is expanding to the scientific domain where there is an increasing number of collaboratories. An example of a social Web community in this domain is the new myExperiment site [26]. Their goal is to enable "scientists to share, re-use and re-purpose their workflows and reduce time-to-experiment, share expertise and avoid reinvention". The medley infrastructure can be integrated with these sites to provide a flexible mechanism for users to combine multiple workflows and services from a large, shared pool. In such a scenario, medleys can also serve as an unobtrusive mechanism for capturing semantics and domain-specific knowledge. For example, when a user synchronizes components from different services, this indicates that these components are related (and compatible). Such knowledge can be re-used to help other users compose new applications.

Biton et al. [27] proposed the creation of views over workflows. Their views are similar to our notion of workflow view. Their objective, however differs from ours in that their goal is to deal with the overload of provenance derived from the workflow runs, by controlling the granularity at which provenance is collected (or published) through these views.

6 Conclusion

Workflow medleys represent a new approach for manipulating ensembles of workflows. Our framework combines a set of operations that are common in exploratory tasks with an intuitive visual interface. We have studied our approach by examining ways it could be applied to different application areas, and have seen that medleys help simplify workflow-based exploratory tasks. We plan to conduct user studies to further evaluate our approach with respect to usability and effectiveness.

Acknowledgments. This work is partially supported by the NSF (under grants IIS-0513692, CCF-0401498, EIA-0323604, CNS-0514485, IIS-0534628, CNS-0528201, OISE-0405402), the DOE, and an IBM Faculty Award. E. Santos is partially supported by a CAPES/Fulbright fellowship.

References

1. : Yahoo! Pipes http://pipes.yahoo.com.
2. : The Taverna Project http://taverna.sourceforge.net.
3. : The VisTrails Project http://www.vistrails.org.
4. Aalst, W., Hee, K.: Workflow Management: Models, Methods, and Systems. MIT Press (2002)
5. : Business process execution language for web services version 1.1. http://www.ibm.com/developerworks/library/specification/ws-bpel (February 2008)
6. J. Freire, C. T. Silva, S. P. Callahan, Santos, E., Scheidegger, C.E., Vo, H.T.: Managing rapidly-evolving scientific workflows. In: International Provenance and Annotation Workshop (IPAW). LNCS 4145 (2006) 10–18 Invited paper.
7. Mitchell, T.: Machine Learning. McGraw Hill (1997)
8. : The matplotlib library. http://matplotlib.sourceforge.net
9. : The Kepler Project http://kepler-project.org.
10. Lee, E.A., Parks, T.M.: Dataflow Process Networks. Proceedings of the IEEE **83**(5) (1995) 773–801
11. : The Chembiogrid web site. http://www.chembiogrid.org
12. Schroeder, W., Martin, K., Lorensen, B.: The Visualization Toolkit An Object-Oriented Approach To 3D Graphics. Kitware (2003)
13. Parker, S.G., Johnson, C.R.: SCIRun: a scientific programming environment for computational steering. In: Supercomputing. (1995)
14. Deelman, E., Singh, G., Su, M.H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G.B., Good, J., Laity, A., Jacob, J.C., Katz, D.S.: Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems. Scientific Programming Journal **13**(3) (2005) 219–237
15. : Microsoft Workflow Foundation http://msdn2.microsoft.com/en-us/netframework/aa663322.aspx.
16. Foster, I., Voeckler, J., Wilde, M., Zhao, Y.: Chimera: A virtual data system for representing, querying and automating data derivation. In: Statistical and Scientific Database Management (SSDBM). (2002) 37–46
17. Lawrence, P., ed.: Workflow Handbook. Workflow Management Coalition. John Wiley and Sons (1997)
18. van der Aalst, W.: Business process management: A personal view. Business Process Management Journal **10**(2) (2004) 135–139
19. Mohan, C., Alonso, G., Günthör, R., Kamath, M.: Exotica: A research perspective ob workflow management systems. IEEE Data Engineering Bulletin **18**(1) (1995) 19–26
20. Deelman, E., Gil, Y.: NSF Workshop on Challenges of Scientific Workflows. Technical report, NSF (2006) http://vtcpc.isi.edu/wiki/index.php/Main_Page.
21. : Flickr http://www.flickr.com.
22. : Facebook http://www.facebook.com.
23. Viegas, F.B., Wattenberg, M., van Ham, F., Kriss, J., McKeon, M.: Many eyes: A site for visualization at internet scale. IEEE Transactions on Visualization and Computer Graphics **13**(6) (2007) 1121–1128
24. : Swivel http://www.swivel.com.
25. : Social data analysis workshop (2008) http://researchweb.watson.ibm.com/visual/social_data_analysis_workshop.
26. : myexperiment http://www.myexperiment.org.
27. Biton, O., Cohen-Boulakia, S., Davidson, S.B.: Zoom*userviews: querying relevant provenance in workflow systems. In: VLDB '07: Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment (2007) 1366–1369