Analyzing the Gap Between Workflows and their Descriptions

Paul Groth and Yolanda Gil

Information Sciences Institute, University of Southern California

pgroth@isi.edu

Abstract

Scientists increasingly use workflows to represent and share their computational experiments. Because of their declarative nature, focus on pre-existing component composition and the availability of visual editors, workflows are often seen as more "natural" than programming or scripting languages for representing data analysis procedures. However, there is still a significant gap between the naturalness of workflow representations and natural language. In this paper, we aim to identify key constructs that intelligent workflow systems could support to allow for more natural workflow representations. These constructs are identified through a comparison of bioinformatics workflows and their associated descriptions obtained from the virtual research environment myExperiment.

1. Introduction

As the use of computational techniques by bench and other "end user" scientists (e.g. those that are not trained formally or otherwise in computer science/programming) intensifies, the need for systems to ease their use increases. Additionally, because science requires the constant modification, combination, and creation of new experimental techniques, such systems most cater not only for the *use* of these techniques but also for their *editing*.

Workflow systems have stepped into this role by providing a means to compose, execute, and manage computational experiments and data analyses. Workflows declaratively capture the steps of an analysis and the dependencies between them [7]. Steps are represented as components (e.g. software programs or web service invocations) that define the computations that should take place.

Because of their declarative nature and their reliance on preexisting components, workflows are often thought of as more natural than other representations such as programs and scripts. Additionally, workflow systems often provide graphical user interfaces for the composition and editing of workflows. Examples include Taverna [13], Kepler [19], and VisTrails [2] as research environments for scientific workflows, additionally commercial workflow editors such as Tivoli and YAWL are available. While the editing interfaces of these systems differ in some respects, they all follow the general approach of representing workflows as a series of nodes (components) and arcs depicting dataflow between components.

While workflow systems, in particular those with graphical user interfaces, have been successfully used by scientists, there is still a significant gap between the workflow representation of an experiment and its description in natural language. By quantifying this gap, valuable insight can be gained into what users consider to be the most pertinent information when naturally describing workflows. In this paper, we measure this gap by comparing the natural language descriptions of bioinformatics workflows with their associated workflow representations by measuring the difference between the procedural information constructs they contain. For example, a workflow representation might specify six detailed steps, whereas the associated description may focus on the two key steps within the workflow. Using this information, we identify key constructs that workflow systems can support to make them more natural. The aim is to provide evidence for the inclusion of certain features in workflow systems.

The rest of the paper is organized as follows. We begin by describing the workflow corpus under consideration. Section 3 then defines the categories of procedural information used in the analysis. Using these categories, Section 4 presents the results of comparing workflow descriptions to their associated concrete workflow representations. Section 5 summarizes the results of this analysis. Suggestions for possible workflow constructs based on this analysis are given in Section 6. Finally, we present related work and conclude.

2. Workflow Corpus

We performed an analysis of workflows from myExperiment (www.myExperiment.org), a virtual research environment that facilitates collaboration and sharing of workflows through a social web approach [3]. Users can upload and retrieve workflows, tag them with annotations, connect to their associates (i.e. "friends"), and message other users. myExperiment also supports the packaging of experimental results and other data with workflows. At the time of this study (Sept 30 2008), myExperiment had 1181 users and 451 workflows. The workflows currently within myExperiment are edited and executed using the Taverna workflow system [13].
An example workflow is shown in Figure 1 with its corresponding textual description. In our analysis, we

include the title of the workflow as part of the description. It is well known that human instruction is often incomplete, erroneous, and out of order [23], and as we will see workflow descriptions are no exception. But because these workflows are executable, the dataflow diagram provides a complete specification of the workflow that the user describes with text. The descriptions provide us with useful examples of how humans describe workflows naturally, while still having a gold standard of what workflow the editor was supposed to create given that information.
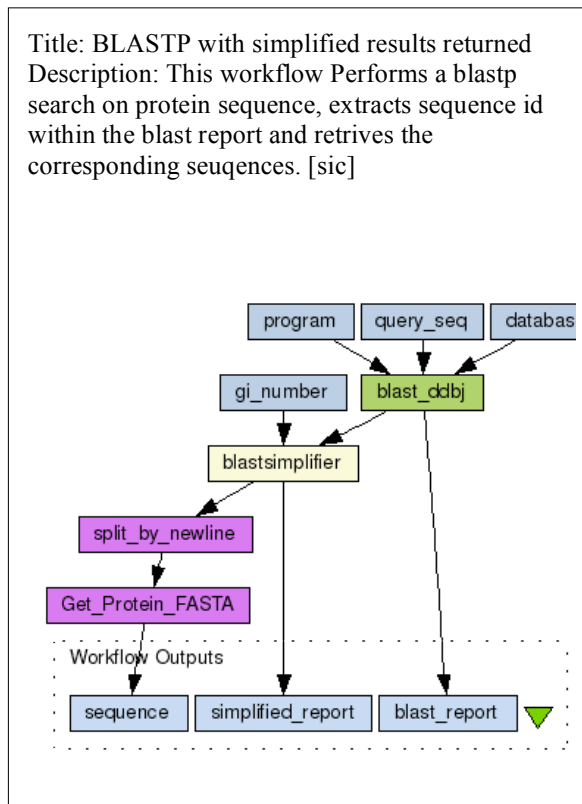
Title: BLASTP with simplified results returned
Description: This workflow Performs a blastp search on protein sequence, extracts sequence id within the blast report and retrives the corresponding seuqences. [sic]

Figure 1: A myExperiment workflow

We selected workflows that **wer**e tagged as "bioinformatics", a domain that we were reasonably familiar with. There were 30 workflows in total, which we used for our analysis. The number of components in the workflows ranged from 1 to 11, the number of dataflow links ranged from 3 to 18. Some scientific workflows can include thousands of jobs when executed, but they have an underlying parallel structure that can typically be described with a high-level pattern of a few components [8].

Our findings are consistent with studies about human instruction of procedures, as we will note throughout the section, which reinforces the results of our analysis since workflows can be seen as procedures. We believe this corpus provides us with an adequate representation of how users would specify workflows in an editor.

3. Categories of Information

We researched the literature on human descriptions of procedural instructions, which included cognitive studies of subjects providing instructions for a variety of tasks [4, 28, 26, 6], natural language interpretation of written procedures [30, 20], and corpus analysis of instructional text [16, 21]. We categorized the kinds of information expressed in several broad categories. We describe relevant categories here, and when relevant the equivalent terminology for workflows is also provided. The categories are:

- Information about *steps* in a procedure. This category includes information about the type of steps or components within the procedure or workflow. For example, in the bioinformatics domain, a common component type is "sequence alignment". Also falling within this category, are the bindings of objects to steps. Bindings map to the inputs of a workflow component. Steps may also have preconditions and effects. In workflows, preconditions are exemplified by the constraints on the input types for components. For example, the statement "this workflow simplifies a BLAST text file" identifies input file as having the type "BLAST text". Information about effects is equivalent to a component's output(s). Finally, instructional texts convey details about step ordering. For example, "perform blast_ddbj before blastsimplifier".

- *Goal statements* provide information about how to interpret procedures essentially describing the intended outcome of following the procedure's steps. In the context of workflow systems, the goal of executing a workflow is often evident in the title of the workflow, for example, the "Fetch PDB flatfile from RCSB server", describes the purpose of using the particular workflow in question.

- *Organizational information* helps the interpretation of complicated procedures by breaking down procedures in to more accessible constituent parts. Organizational information is often expressed using a goal decomposition schema, which models a procedure as a series of goals that can in turn be decomposed into a series of sub-goals. Concretely, in a workflow, this is the ability for a workflow to call out to or contain other sub-workflows.

- Information about procedures can also be categorized as *advice* (both positive and negative) about how to perform when confronted with different options and deal with exceptions to a general procedure. An example from a workflow description is "Please use two blast text files inputs for a secure result."

Other categories that appear in natural human instruction include background, device models, structural and functional organizational knowledge, situated instructions, and examples. We do not discuss them here because they

Type of information	Workflows containing this type of information in their text description	Workflows containing all assertions of this type in their text description	Workflows missing some in their text description	Amount of individual assertions of this type present in the text description			Amount missing in the text description of this type that appears in the diagram		
				median	min	max	median	min	max
Component type	29 (96.6 %)	8	21 (66.6%)	3	0	11	2.5	0	9
Component inputs	24 (80 %)	6	18 (60 %)	1	0	4	1	0	4
Component outputs	19 (63.3 %)	2	17 (56.6 %)	1	0	4	1	0	8
Component input types (preconditions)	22 (73.3 %)	- *	- *	1	0	4	-	-	-
Component Ordering	11 (68.7 %) **	6 **	5 (31.2 %)**	1.5	0	8	0	0	3
Goals	28 (93.3 %)	-	-	1	0	1	-	-	-
Sub-workflows	1 (3 %) ***	0	12 (40 %) ***	1	1	1	0	0	4
Advice	6 (20 %)	-	-	0	0	2	-	-	-
(-) Denotes not applicable. * Preconditions are not represented in the workflow diagram ** Only among the 16 workflow descriptions that had more than one component. Only orderings pertaining to components in the description are counted. *** Only among the 13 workflows that contained sub-workflows.									

Table 1: Analysis of types of information in textual workflow descriptions

did not appear frequently in the workflow descriptions we analyzed.

4. Categorizing Information in Workflow descriptions

Table 1 shows the results of determining the categories of information found in workflow descriptions. In addition to this categorization information, the results show the discrepancies between the textual descriptions and what is specified in the workflow itself. To aid understanding, the types of information in the table are referred to using workflow terminology. We now address each information category in turn. Note that there were five assertions in the corpus that did not fall into these categories.

Step Information

The most prevalent type of information found in the workflow descriptions was related to describing components. In particular, 96.6% of the descriptions contained some information about component types. However, the descriptions rarely contained all the components that appeared within the workflow. Only eight workflow descriptions contained all the components. The median number of components not mentioned for the other workflows was 2.5.

Users mention in their text descriptions only the most important components within the workflow. This is consistent with studies of human instruction, where important steps are mentioned explicitly while unimportant steps are implicit [4, 6]. For example, in the workflow with the maximum amount of missing component types, of the 9 missing components, seven were conversion components designed to take the output of one component and convert it to another format for consumption by other components or the end user. In some cases, the need for these format conversions could be inferred from the descriptions. This would mean that a system could automatically hypothesize format conversion components that are not mentioned by users. However, we note that in other cases there are 3 or 4 conversion components that could not be inferred from the description alone.

Workflow steps are often described first in an abstract manner and then described again with a more concrete type. We found 18 workflows where this occurred. Iterating over the steps may be a form of emphasizing the importance of the steps. Take for instance the following description from the workflow "Protein_search_fetch_align_tree":

> *"Description: An implmentation of the classical sequence analysis workflow:*
> *1. Find homologues (sequence similarity search)*

The workflow author first lists the components with abstract types sometimes even providing an additional type for clarity in parentheses and then relists the components with their more concrete type.

The same focus on critical items is also evident for input and output information. Inputs to components were found in 80% of the workflow descriptions. However, in only six cases were the inputs fully specified. In another 18 cases, only partial inputs were provided, which seemed to highlight the important inputs. For example, in Figure 1, the description fails to mention the ancillary inputs of program and database. On the other hand, in this corpus, only one input is generally missing (median 1) and from our observation it seems that picking reasonable defaults could set many of the missing inputs.

As well as specifying what are the inputs of each component, 73.3% of the descriptions provided information about the type of those inputs. The types of inputs ranged from specifying simply that an input is an "identifier" to more complex types such as "an input sequence alignment of homologues". We did not collect data about the specificity or adequacy of the types.

Output information was found in 63.3 % of the descriptions and important outputs were emphasized such as the phylogenic tree mentioned in step four of the previous example description. The almost 20% gap between the number of workflow descriptions with output descriptions and those with input descriptions is supported by the literature as humans often neglect to mention effects in instruction [20]. Outputs, like inputs, were also given types. It is interesting to note that there are outputs in workflows such as logs or provenance information that are not core to users goals of producing a scientific result but are of interest to others (e.g. reviewers, funding agencies). Thus, the lack of their mention in the descriptions may be the result of the particular perspective of the user.

The analysis of component ordering (i.e. step ordering) only applies to the workflow descriptions with more than one component. Out of those sixteen descriptions, only roughly a third did not have all of the required component orders. One of the descriptions specified the wrong order for two components. Note, in this analysis, we did not quantify the difference between the component orderings in the description and in the workflow for those components that were only represented in the workflow.

Goal Information

The goal statement or purpose of a workflow is predominately given in the title of the workflow. 93% of the workflows had title that identified the purpose of the workflow. An example of a title that did not convey a goal was "Genome annotation pipeline demonstrator for nucleic acids research." Goal statements have been found to be useful to interpret, recall, and reuse human instruction [29].

Sub-Workflows

Most workflows contained no explicit organizational information. However, the workflows themselves often contained sub-workflows (i.e. goal decompositions schemas). 40% of the workflows had embedded workflows. It seems that users refer to already defined workflows just like they refer to components with no distinction. For example, "multiple sequence alignment" refers to a complex workflow and "sequence alignment" refers to a component.

Advice

In 20% of the descriptions, both positive and negative advice was present for describing how other users should use the described workflow. An example of negative advice is "N.B. this workflow does not function correctly as it is designed for use with NCBI blast scripts". An example of positive advice from a workflow to perform BLASTP was "Please use two blast text files inputs for a secure result".

Vocabulary

The terms used to describe components were almost always different in the text description than the term used in the workflow. In only four cases did all the component names found within the description, appear within the workflow. For 86% of the workflow descriptions, there was a mismatch between the component names used and those that appeared within the workflow. An example of this mismatch is shown in Figure 1, "blast_bbdj" in the workflow is referred to as "blastp search" in the description.

5. Summary of the Analysis

Using these results, we come to the following conclusions. Users specify key components used in a workflow. However, they often leave out components assuming their

inclusion can be inferred. Likewise, users specify important bindings (inputs) and effects (outputs) including their types, while leaving many necessary inputs and outputs unspecified. Generally, descriptions contain a correct ordering for those steps found within the description. While many workflows contained sub-workflows, the descriptions did not describe these decompositions. Instead, users treated workflows as just another component that could be referred to by name. Advice was given to indicate when to use the workflow, and appeared both in positive and negative forms. Finally, the vocabulary used in the textual descriptions differs dramatically from the component names used within the workflows.

6. Constructs for Natural Workflow Editing

This analysis leads us to the following suggested constructs for the creation of workflow systems that are increasingly natural.

1) Allow for variable vocabulary

Vocabulary variability, where users use different terms to refer to the same entity, is a well-known problem when dealing with free text input from users. [5] found that less than a dozen people out of a thousand would use a term that had been pre-selected to refer to a specific computer command. This problem was also evident in our analysis.

One solution to vocabulary variability is to restrict users to a predefined set of terms to refer to components and data. This is clearly unnatural as users continually use the vocabulary that fits best to their situation. Instead, we suggest two mechanisms for allowing flexible vocabulary: search and personalized synonym dictionaries.

In recent work, a tool for finding API components (a similar environment to workflow systems), text based search over the entire Internet was shown to be an effective mechanism to allow programmers to find the correct component using their own vocabulary [30]. In addition, free text search is becoming fundamental for workflow construction, as the number of available components and data sets has exploded. For example, there are over 1000 different components available for constructing bioinformatics workflows [18]. With so many components, finding the correct one through browsing trees or using menus is untenable.

In addition to search, we suggest the introduction of personalized synonym dictionaries. Work on understanding users To-Do lists, shows that they are highly personalized [9]. Similarly, the vocabulary used to refer components and data is specialized to a user and their particular field. For example, a biologist might use the term sequence alignment whereas a technician might use the term blast_p. As the user constructs a workflow with search, the system can

keep track of how search terms were used to find components building a personalized dictionary of synonyms. Later, these synonyms could be used directly in a workflow as references to components.

2) Automatic introduction of conversion components

In their descriptions of workflows, users focus on critical operations ignoring those that are peripheral to the main task. However, by necessity workflows often contain large numbers of components that manipulate data so that it can be feed from one component to another. Because these conversion components (or shims) are not part of the experimental procedure, their introduction should be automatic and their presence hidden from the end user.

There is ongoing work in semantic web services and in automated goal-directed construction of workflows that could be leveraged to facilitate this feature [1].

3) Automatic introduction of inputs and outputs

Just as with components, users focus on the important data within their workflow. Workflow systems can easily begin to allow users to focus on specific data sets through the use of *reasonable defaults* for component inputs. Many component developers already specify such defaults for parameters within documentation. For example, the default parameters of the Blast sequence alignment component filter out low complexity regions of sequences[1]. For other inputs where the component description does not define a default, it may be useful to mine existing usage of the component to find plausible defaults. Furthermore, it would also be helpful for the interface to signal what inputs the user needs to specify.

Systems can additionally present users possible options for particular inputs. A good example of this is the Wings workflow system [8], which relies on component registries and data catalogs to fill in possible data values for inputs. Using Wings, the user can then select from a variety of workflows with prefilled inputs.

In terms of outputs, they are often not mentioned in typical human instruction. Relying on component descriptions within registries is one route to providing the kind of output information workflow systems require. It is interesting to note that most workflow systems emphasize data flow in their graphical interfaces and thus both output and input specification are equally emphasized. A future direction of work would be to develop interfaces that highlight inputs over outputs. An example of such a system is Kepler's

---

[1] From NCBI Blast rules of thumb (http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/rules.html accessed Feb. 9 2009).

pipeline model [19], where components are configured to modify a stream of data and do not specify specific outputs.

4) Workflows as sub-procedures
A key result of our analysis is that users make no distinction between workflows and components. They refer to workflows using similar terms as they would any other component. Several workflow systems allow for workflows to be called from other workflows including Taverna, the system used to construct the workflows in our study. To make these calls as natural as possible, workflows should be exposed in the same manner as components in the interface. This requires that workflows have similar interface definitions to components. For example, if a component specifies its inputs and outputs so should the workflow definition. In summary, the ability to analyze the definition of a procedure should be separate from the ability to invoke it.

5) Transparent collection of provenance information

Another ramification of users' focus on the critical elements of their computational experiments is the absence of definitions for workflow elements that may be important to other stakeholders besides the user. For example, the log file outputs of a particular component may be of use for reviewers validating an experiment but not to the user themselves. To address this inadequacy, workflow systems should provide for the transparent collection of provenance information. Fortunately, this feature is present in many different workflow systems and is an active area of research within the community [23].

In addition to the capturing provenance, workflow systems should provide mechanisms that make the user aware of other stakeholder's needs during the editing process. For example, by displaying the requirements of different users roles or by providing templates for workflows that specify what different stakeholder's may require. When describing procedures in natural language, users omit information that they believe their intended target audience already knows, the assistance suggested here will help users be aware of other audiences that may not have the appropriate background knowledge helping the user to create more reusable workflows that can be verified and checked by others.

6) Direct encoding of advice and goals

In 20% of the workflows in our analysis, the descriptions contained advice about how to use the workflow. In the case of the myExperiment workflows we studied, this advice was not encoded in the concrete representation. Thus, we suggest that advice should be explicit in the workflow representation. For example, users should be able to encode whether they found a workflow to be useful. Additionally, the creator of a workflow should be able to

identify particular data sets that work well with the workflow. Both mechanisms could be used to provide automated assistance to users of workflows.

In addition to advice, workflow interfaces should provide a means to encode goals into a workflow. 93.3% of the workflow descriptions contained goals and these were not present in the workflows themselves. To encode goals, a workflow system could provide the ability to mark a particular output as the goal of the workflow. Likewise, a workflow system could allow the user to provide tags that describe the workflow's goals. These "goal-tags" would enable users to search for workflows that satisfied their particular need.

7) Use visual-textual hybrid interfaces

As discussed previously, the predominant interface for the construction of workflows is a visual editor operating on a graph of dataflow between components. Given that the textual descriptions of workflows provide additional knowledge and semantics beyond what is represented in the workflow itself, it may be beneficial to investigate natural language interfaces in combination with the standard visual construction interfaces. This is particularly appropriate given that instruction using natural language is a very common way to describe procedural artifacts [20, 21].

Additionally, research into visual programming calls into question the superiority of visual notations over textual languages [24]. For example, Green et al found that on a sample of programming tasks, users performed better with a textual language than a graphical one [12]. Visual languages also require that users "think ahead" so that the program maintains a readable layout as the program is modified [11]. Finally, when comparing program comprehension in graphical and textual languages, users had similar performance [22]. This is not to say that visual interfaces are not effective. Instead, we believe a hybrid approach that combines visual and textual input could provide a more natural interface. Such an approach has been found to be very effective in other areas [15, 24, 27].

While some existing workflow systems support a subset of the constructs discussed above, we hope that this review helps workflow system developers in defining their feature roadmap and reinforces design decisions that are congruent with our analysis. We believe that by extending workflow systems to add these constructs workflow editing will become more natural for the end-user scientist.

7. Related Work

Workflow systems abound, instead of focusing on these technologies here, we discuss studies of workflow artifacts as well as end-user procedural knowledge capture

environments that may not be as familiar to those in the scientific workflow community.

[10] studied the myExperiment workflow repository to gain insight into how scientists shared and reused workflows. The work combined an analysis of myExperiment website logs with a survey user scientists. In contrast, our analysis focuses on workflow representation not their sharing. Additionally, we use a different data set namely the workflows retrieved from myExperiment.

The provenance of a workflow's evolution as the user constructs it can also be used to determine how users interact with a workflow system. [17] describes tracking a workflow's evolution and what metrics are indicative of a particular usage patterns. For example, the branching factor of the provenance graph can be used to determine how much trial-and-error a user is engaging in. The study used workflows captured from 30 students in a visualization class. This work could be used to verify that the constructs suggested here are indeed useful.

Visual interfaces and other natural interface designs have been developed that are effective means for end users to specify procedural information to a system. [14] provides a thorough overview of such systems, which they call empowering systems since they aim to help non-programmers to specify behaviors for a computer system. Some of the best-known systems are AgentSheets, Stagecast, Logo, Alice, Forms/3, and Hypercard. Successful techniques include programming by rehearsal by personifying components, the use of domino icons or comic strips to show before and after states for actions, 2-D grids and patterns for specifying conditions for behavior rules, physical metaphors to represent objects and behaviors, associating behaviors to interface components, object-centric commands that can be interpreted, commands with graded complexity in parameters, aggregate operations over objects, making specifications alive so they can always be tested even if partially specified, extending spreadsheets to create new data types and their associated behaviors, event-triggered behaviors, and visual dataflow languages. Most of these tools are designed to target specific tasks, objects, or behaviors. They have been shown effective in experiments with non-programmers, and some are commercialized and have been used by thousands of users. Perhaps integrating some of these approaches into workflow interfaces would be beneficial.

8. Conclusion

In this work, we presented a comparison of the information contents of publically available workflows and their associated descriptions. Based on this analysis, we presented 8 workflow constructs for creating natural workflow editing interfaces. To investigate these recommendations, we have begun to extend the Wings

Workflow System to support the constructs above. Specifically, we have added a command line to the interface to create a visual-textual hybrid that supports the entry of natural language.

In the future, we aim to study a wider range of workflows from different domains. Additionally, we would like to contrast computational workflows to written peer-reviewed scientific protocols like those seen in Nature Protocols. Finally, we aim to evaluate the suggested workflow constructs in human studies as well as using heuristic evaluation approaches [25].

Acknowledgements

9. References

1. Ambite, José Luis, Kapoor, Dipsy. Automatically Composing Data Workflows with Relational Descriptions and Shim Services. Proceedings of the 6th International Semantic Web Conference (ISWC-2007), Busan, Korea, November 2007.
2. Callahan S.P., Freire J., Santos E., Scheidegger C.E., Silva C.T., and Vo H.T. "VisTrails: Visualization meets Data Management." In Proceedings of ACM SIGMOD 2006.
3. De Roure, D., Goble C., Stevens, R. "The design and realization of the myExperiment Virtual Research Environment for social sharing of workflows". Future Generation Computer Systems. In Press.
4. Dixon, P., Faries, J., and Gabrys, G. "The role of explicit action statements in understanding and using written directions." *Journal of Memory and Language*, 27, 1988.
5. Furnas, G., Landauer, T., Gomez, L., and S. Dumais. "The Vocabulary Problem in Human-System Communication." *Communications of the ACM*, 30, 1987.
6. Galotti, K. M and W. F. Ganong. "What Non-Programmers Know about Programming: Natural Language Procedure Specification." *International Journal of Man-Machine Studies*, Vol 22, 1985.
7. Gil Y, Deelman E., Ellisman M., Fahringer T., Fox G., Gannon D., Goble C., Livny M., Moreau L., Myers J., "Examining the Challenges of Scientific Workflows," Computer , vol. 40, no. 12, pp. 24-32, December, 2007.
8. Gil Y., Ratnakar V., Deelman E, Mehta G, and Kim J. "Wings for Pegasus: Creating Large-Scale Scientific Applications Using Semantic Representations of Computational Workflows," *Proc. of the 19th Annual Conf. on Innovative Applications of Artificial*

Intelligence (IAAI), Vancouver, British Columbia, Canada, July 22-26, 2007.

9. Gil Y. and Ratnakar, V. "Automating To-Do Lists for Users: Interpretation of To-Dos for Selecting and Tasking Agents." Proc. of the Twenty-Third Conference of the Association for the Advancement of Artificial Intelligence (AAAI-08), Chicago, IL, July 13-17, 2008.

10. Goderis, A., De Roure, D., Goble, C., Bhagat, J., Cruickshank, D., Fisher, P., Michaelides, D. and Tanoh, F. (2008). Discovering Scientific Workflows: The myExperiment Benchmarks. Submitted for publication.

11. Green, T.R.G. and M. Petre (1996). "Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework." Journal of Visual Languages and Computing 7(2): 131-174.

12. Green, T.R.G., M. Petre and R.K.E. Bellamy (1991). "Comprehensibility of Visual and Textual Programs: A Test of Superlativism Against the 'Match- Mismatch' Conjecture." Empirical Studies of Programming: Fourth Workshop. J. Koenemann-Belliveau, T. G. Moher and S. P. Robertson.

13. Hull D, Wolstencroft K, Stevens R, Goble C, Pocock MR, Li P, Oinn T. "Taverna: a tool for building and running workflows of services." Nucleic Acids Res., 34, 2006.

14. Kelleher, C. and R. Pausch. "Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers." *ACM Computing Surveys*, 37(2), 2005.

15. Koedinger, K. and J. Anderson. "Abstract Planning and Perceptual Chunks: Elements of Expertise in Geometry." *Cognitive Science*, 14(4), 1992.

16. Kosseim, L. and G. Lapalme. "Choosing Rhetorical Structures to Plan Instructional Texts". *Computational Intelligence*, 16(3), 2000.

17. Lins, L., Koop, D. Koop, Anderson E., Callahan S.P., Santos E., Scheidegger C.E., Freire J., and Silva C.T. Examining Statistics of Workflow Evolution Provenance: A First Study. In Proceedings of International Conference on Scientific and Statistical Database Management (SSDBM), 2008.

18. Lord P., Alper P., Wroe C., and Goble C. "Feta: A lightweight architecture for user oriented semantic service discovery." 2nd European Semantic Web Conference, (2005)

19. Ludäscher B., Altintas I., Berkley C., Higgins D., Jaeger-Frank E., Jones M., Lee E., Tao J., Zhao Y. "Scientific Workflow Management and the Kepler System." *Concurrency and Computation: Practice & Experience*, 18(10), pp. 1039-1065, 2006.

20. Mahling, D.E. and Croft, W.B. 1988. "Relating human knowledge of tasks to the requirements of plan libraries." International Journal of Human-Computer Studies, Vol. 31.

21. Miller, L. "Natural language programming: Styles, strategies and contrasts", *IBM Systems Journal*, 20, 1981.

22. Moher, T.G., D.C. Mak, B. Blumenthal and L.M. Leventhal (1993). "Comparing the Comprehensibility of Textual and Graphical Programs: The Case of Petri Nets." Empirical Studies of Programmers: Fifth Workshop. C. R. Cook, J. C. Scholtz and J. C. Spohrer.

23. Moreau, L. Plale B., Miles S., Goble C., Missier P., Barga R., Simmhan Y., Futrelle J., McGrath R., Myers J., Paulson P., Bowers S., Ludaescher B., Kwasnikowska N., Bussche J., Ellkvist t., Freire J., and Groth P. The Open Provenance Model (v1.01). Technical report, University of Southampton, July 2008.

24. Nardi, B. A. (1995). "A Small Matter of Programming: Perspectives on End User Computing." MIT Press, Cambridge, MA, 1995.

25. Nielsen, J. (1994). "Heuristic Evaluation. Usability Inspection Methods." J. Nielsen and R. L. Mack. New York, John Wiley & Sons: 25-62.

26. Nyberg, E. and T. Mitamura. "Controlled Language and Knowledge-Based Machine Translation: Principles and Practice." Proceedings of the First International Workshop on Controlled Language Applications (CLAW), 1996.

27. Pane, J. F. & Myers, B. A. (1996), 'Usability Issues in the Design of Novice Programming Systems', Technical report, Carnegie Mellon University.

28. Smith, D. C., Cypher, A., Tesler, L. G. 2000. "Novice Programming Comes of Age". *Communications of the ACM* 43(3).

29. Steehouder, M., Karreman, J., and N. Ummelen. "Making Sense of Step-by-Step Procedures". Proc of the 18th annual ACM Int. Conf. on Computer Documentation:Ttechnology & Teamwork, 2000.

30. Stylos J. and Myers B. (2006) "Mica: A Programming Web-Search Aid". IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2006).