http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (15): 3424-3429, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.3424.3429

An Approach to Scientific Workflow Slicing

Mengmeng Yang, Jie Cheng and Wei Xu
School of Mechanical, Electrical and Information Engineering, Shandong University,
Weihai, 264209, Shandong, China

Abstract: Scientific workflow technology has been a useful tool in scientific computations. Since, scientific workflows are normally data-intensive and have complex structures, how to find the relevant knowledge that scientists concern about in a large scientific workflow is a key issue. This study presents a concept of scientific workflow slice which is a sub-workflow relative to a specific slice criterion and proposes a slice generation algorithm. In addition, we also give the concepts of forward workflow slice and backward workflow slice and propose the slicing methods. Our approach can be seemed as a kind of workflow data filter which is expected to apply in scientific workflow provenance, fault-tolerance and reuse.

Key words: Scientific workflow, workflow slice, slice criterion

INTRODUCTION

Scientific workflow (Romano, 2008; McPhillips *et al*., 2009) is a mechanism to map scientific application to diverse sources, automatically managing the data and resources distributed in different areas. By means of scientific workflow system, scientists do not need to understand the details of underlying implementation but only the workflow specification. Up to now, there have been many researches (Deelman *et al*., 2006; Elmroth *et al*., 2010; Lim *et al*., 2011) about scientific workflows. Normally, scientific workflows are data-intensive and have complex structures (Goble and De Roure, 2009; Callaghan *et al*., 2010). For instance, in a Montage workflow (Berriman *et al*., 2004; Katz *et al*., 2010) which is an astronomy application that assembles astronomical images into mosaics, a mosaic of 6-degree square involves 1,444 input images, requires 8,586 computational steps and generates 22,850 intermediate data products. For such a scientific workflow, scientists normally concern only about some specific functions, thus they will pay attention to part of relevant jobs or data instead of the whole workflow. If they can conduct their experiments or computations based on the relevant sub-workflow, their research will be more targeted and efficient. Hence, how to refine and decompose a scientific workflow relative to the users' interests is a key problem. Inspired by program slicing (Weiser, 1984; Binkley and Gallagher, 1996) which is a well-known decomposition technique to extract all program statements relevant to variables of user's interests, this study proposes a concept of workflow slice that is a subset of a workflow data and jobs that users concern about. However, although the idea of program slicing gives a way to address our problem, workflow slice is quite different from the program slice. For example, a program slice is based on program language consisting of variables and statements, so it is defined as a set of program statements. Whereas, a workflow is composed by data and jobs, thus the workflow slice has different slice criterion and granularity from the program slice. In this study, we give the workflow slice definition and propose a slice-generation algorithm. Besides, we also give the concepts of forward workflow slice and backward workflow slice that will be useful in the research of scientific workflow fault-tolerance and provenance.

In this study, we describe a workflow specification based on DAX (Pegasus Workflow Management System, 2013) which is a useful workflow description language in a form of a Direct Acyclic Graph (DAG) with XML format. It describes the structure of a workflow that can be read by the workflow engine. To illustrate our proposed concepts and algorithm, we use Black Diamond workflow (Pegasus Workflow Management System, 2013) and Montage workflow as the description cases.

The rest of the study is organized as follows. Section 2 defines the workflow slice criterion and presents the workflow slice concept. The slice-generation algorithm is also proposed in this section. In section 3, we take a Montage workflow as the instance to illustrate workflow slice. Section 4 gives the related works and concludes the study.

Corresponding Author: Jie Cheng, School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai, 264209, Shandong, China

WORKFLOW SLICING

Workflow slice concept: Normally, a scientific workflow can be graphically represented as a Directed Acyclic Graph (DAG), where vertexes and directed arcs, respectively represent the jobs and the dependence relationships among them. In such a DAG, data is ignored. While in a scientific workflow, data has become most critical. For illustrative purposes, we extend the scientific workflow definition as follows.

Definition 1: A scientific workflow W is described as W = (D, J, R), where D = $\{d_i|i = 1\text{-}n\}$ denotes the set of data which includes input data, intermediate data and output data, i is the unique identify number of data d; J = $\{j_k|k = 1\text{-}m\}$ denotes the set of jobs to be executed, k is the unique identify number of job j; R = $R_{dj} \cup R_{jd}$ expresses the set of relations, where, R_{dj} = {<d, j>|d$\in$D and j$\in$J} is the set of relations from a data to a job and R_{jd} = {<d, j>|j$\in$J and d$\in$D} is the set of relations from a job to a data.

According to above definition, a workflow DAG can be described as a directed acyclic graph with two types of vertexes and two types of arcs. That is, data vertexes, job vertexes, data-job arcs and job-data arcs. In this study, this type of DAG is called as a job-data DAG. Figure 1 shows the traditional DAG and job-data DAG of a simple Black Diamond workflow (Pegasus Workflow Management System, 2013), from which, we can see that a Black Diamond workflow contains 4 jobs which are "preprocess", "findrange1", "findrange2" and "analyze". The function of the jobs is to read input files designated by arguments, write them into output files and produce some summary information of where and when it will be run.

Basically, workflow slice is a method to extract jobs and data relevant to the users' interests. The idea is inspired by program slicing. Since, a program consists of variables and statements, its slice criterion is a set of variables at some program points of users' interests. Whereas, different from a program, a scientific workflow consists of data and jobs, thus a workflow slice criterion should be defined as a data or a job. Normally, data is more critical in a scientific workflow. So, we define a workflow slice criterion only with data in this study. A workflow slice criterion with a job can be defined in a similar way.

Definition 2: Given a scientific workflow W, a scientific workflow slice Ω is defined as Ω = (A, c), where, A = (α, β, γ) is a sub-workflow in which $\alpha \subseteq$E.D, $\beta \subseteq$W.J, g$\subseteq$W.R; c$\in$W.D is the workflow slice criterion.

Definition 3: A forward workflow slice Ω_f is a sub-workflow in which data and jobs will affect the slice criterion. A backward workflow slice Ω_b is a sub- workflow in which data and jobs will be affected by the slice criterion.

These two concepts can be well used in fault-tolerant and data provenance in scientific workflows. For example, if there is an error in output data after execution, users can trace the error in the forward workflow slice. While if an input data or an intermediate data has to be altered before execution, users can predict the execution result by just paying attention to the backward workflow slice instead of the whole workflow.

Workflow slice-generation algorithm: Given a workflow W and the slice criterion Ω.c, the workflow slice Ω.A can be computed in two steps: Firstly, compute the forward

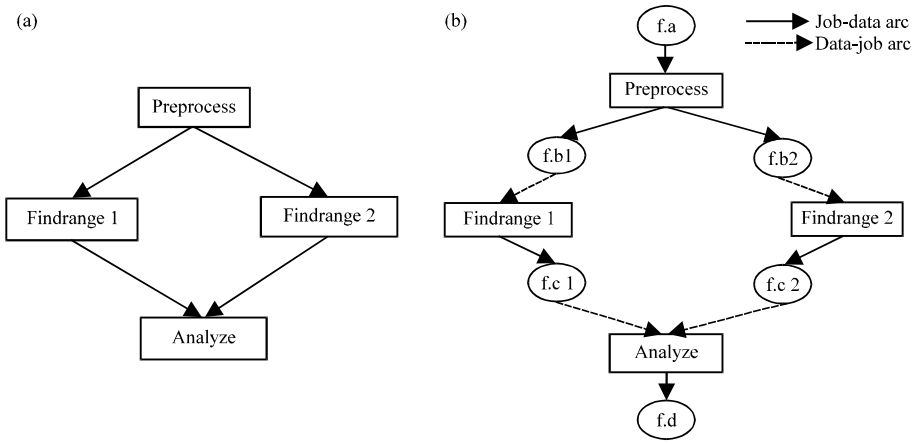

Fig. 1(a-b): Two types of DAG of black diamond workflow (Pegasus Workflow Management System, 2013) (a) DAG of black diamond workflow and (b) Job-data DAG of black diamond workflow

slice relative to the criterion data and then compute the backward slice. The workflow slice generation algorithm is based on job-data DAG which is described as follows:

Input: A scientific workflow W = (D, J, R), input data $D_{input} \in D$, output data $D_{output} \in D$ and a workflow slice criterion Ω.c
Output: A workflow slice Ω.A (α, β, γ)

```
Begin
    Initial α = {c}, β = Φ and γ = Φ;
    Initial a temporary job set s_job = Φ, data set s_data = Φ;
    Forward(c);
    Backward(c);
End;

Forward(c)
{s_data = s_data ∪ {c};
 While(s_data ≠ Φ) Do
 {s_job = Φ;
 For (each d∈s_data )
        For (each <job, data>∈R_jd)
        If (data_id(d) = data_id(data))
                {s_job = s_job ∪ {job};
                γ = γ ∪ {<job, data>};
                s_data = s_data - {d};
                }
        For(each d∈s_data)
        If (d∈D_input) s_data = s_data - {d};
        β = β ∪ s_job;
        For (each j∈s_job)
        For (each<data, job>∈R_dj)
                If (job_id(j)=job_id(job))
                {s_data = s_data ∪ {data};
                γ = γ ∪ {<data, job>};
                }
        α = α ∪ s_data;
}
}

Backward(c)
{s_data = s_data ∪ {c};
 While(s_data ≠ Φ) Do
 {s_job = Φ;
 For (each d∈s_data )
        For (each <data, job>∈R_dj)
        If (data_id(d) = data_ id (data))
                {s_job = s_job ∪ {job};
                γ = γ ∪ {<data, job>};
                s_data = s_data - {d};
                }
β = β ∪ s_job;
 For (each j∈s_job)
        For (each <job, data>∈R_jd )
        If (job_id(j) = job_id( job))
 {s_data = s_data ∪ {data};
        γ = γ ∪ {<job, data>};
        }
        α = α ∪ s_data;
 For (each d∈s_data)
        If (d∈D_output) s_data = s_data - {d};
 }
 }
```

In the algorithm, data_id(d) and job_id(j), respectively represent the identify number of data d and identify number of job j. Figure 2 shows the job-data DAG of the Black Diamond workflow slice with the slice criterion c= f.b2, from which we can see that a workflow slice is a sub-workflow related to the slicing criterion.

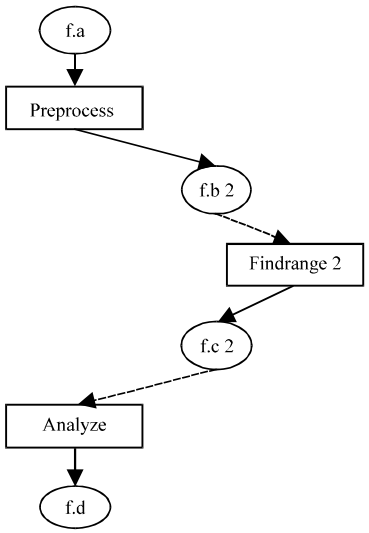

Fig. 2: Workflow slice of a Black diamond workflow with slice criterion c = f.b2

In this study, we take use of DAX as the workflow specification to illustrate the workflow slicing algorithm. The DAX of a Black Diamond workflow (Pegasus Workflow Management System, 2013) is as follows:

```
<!-- Part 1: Files Used -->
<filename file="f.a" link="input"/>
<filename file="f.b1" link="inout"/>
<filename file="f.b2" link="inout"/>
<filename file="f.c1" link="inout"/>
<filename file="f.c2" link="inout"/>
<filename file="f.d" link="output"/>
<!-- Part 2: Definition of Jobs -->
<job id="j1" name="preprocess">
  <argument>-a preprocess -T 60 -i <file name= "f.a"/> -o <file
name="f.b1"/> <file name="f.b2"/> </argument>
</job>
<job id="j2" name="findrange1">
  <argument>-a findrange -T 60 -i <file name= "f.b1"/> -o <file
name="f.c1"/></argument>
</job>
<job id="j3" name="findrange2">
  <argument>-a findrange -T 60 -i <file name= "f.b2"/> -o <file
name="f.c2"/></argument>
</job>
<job id="j4" name="analyze">
 <argument>-a analyze -T 60 -i <file name="f.c1"/> <file name="f.c2"/> -o
<file name="f.d"/> </argument>
</job>
<!-- Part 3: Control-Flow Dependencies -->
<child ref="j2">
 <parent ref="j1"/>
</child>
<child ref="j3">
 <parent ref="j1"/>
</child>
<child ref="j4">
 <parent ref="j2"/>
 <parent ref="j3"/>
</child>
```

As shown above, a DAX file consists of three parts. The first part includes all data in the workflow, in which "input" "inout" and "output" represents the input data, intermediate data and final output data, respectively. The second part defines all jobs of the workflow. Besides, it displays the dependencies between jobs and data. There are four jobs in the DAX file of Montage workflow. For each job, "-a", "-i" and "-o", respectively represent the input parameters, input data and output data. The third part of the DAX file is the dependencies between jobs.

In a Black Diamond workflow, given a slice criterion c = f.b2, based on the slicing algorithm, the forward slice is α = {j1}, β = {f.a, f.b2} and γ = {<f.a, j1>, <j1, f.b2>}. In a similar way, we can get the backward slice as α = {j1, j3, j4}, β = {f.a, f.b2, f.c2, f.d} and γ = {<f.a, j1>, <j1, f.b2>, <f.b2, j3>, <j3, f.c2>, <f.c2, j4>, <j4, f.d>}. As shown in the DAX above, the slice mapped into the workflow specification is described in black text.

A SCIENTIFIC WORKFLOW SLICE CASE

Here, we take a Montage workflow (Berriman *et al*., 2004; Katz *et al*., 2010) as the instance to display the forward workflow slice and backward workflow slice. Montage is a toolkit for assembling astronomical images into mosaics. Montage workflows are normally data-intensive with complex structures. For convenience of description, this section takes a simple Montage workflow case which includes three basic steps that is, image re-projection, background modeling and rectification and co-addition. Figure 3 shows the structure of the simple Montage workflow, where the job "mProject" executes to perform the first step, job "mDiffFit", "mBgModel" and "mBackground" complete the second step and job "mAdd" is for the last step.

Given a workflow slice criterion c, the forward and backward workflow slice can be computed by the slice-generation algorithm. The way of the forward workflow slice is to traverse the graph from the criterion data to the input data. All data and jobs in the traversing path make up the forward slice. Similarly, all data and jobs in the path from the slice criterion to the final output data form the backward slice. Figure 4-5, respectively show the forward workflow slice with the criterion c = 12.txt and the backward workflow slice with the criterion c = correct.tbl. The mapping DAX file of these slices is as follows:

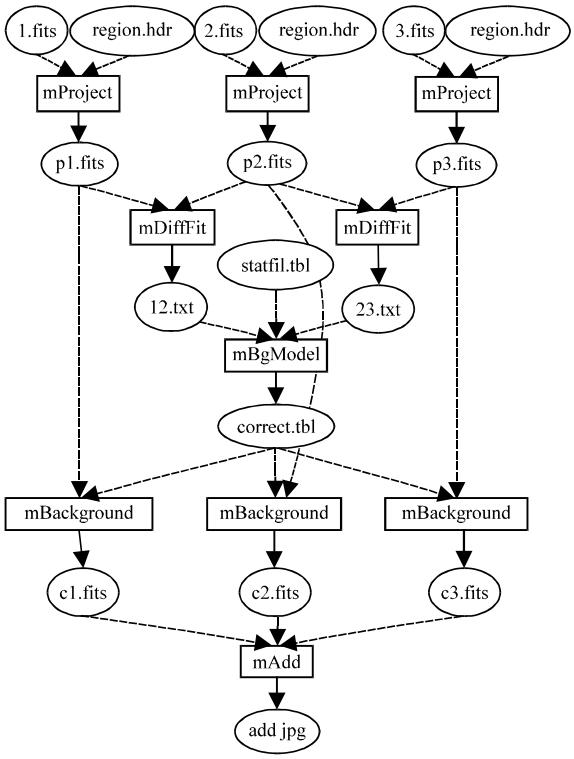

Fig. 3: Job-data DAG of a montage workflow

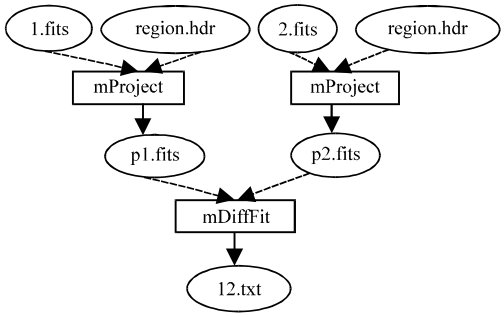

Fig. 4: A forward workflow slice of a montage workflow

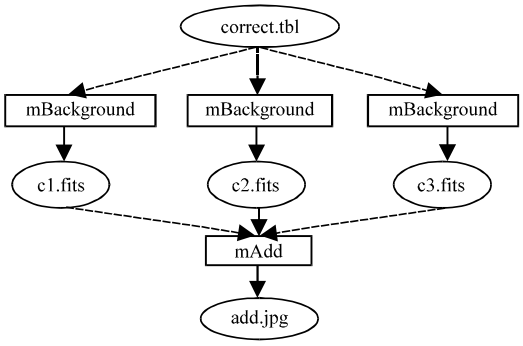

Fig. 5: A backward workflow slice of a montage workflow

```
<!-- Part 1: Files Used -->
<filename file="region.hdr" link="input"/>
<filename file="1.fits" link="input"/>
<filename file="2.fits" link="input"/>
<filename file="3.fits" link="input"/>
<filename file="p1.fits" link="inout"/>
<filename file="p2.fits" link="inout"/>
<filename file="p3.fits" link="inout"/>


<filename file="12.txt" link="inout"/>
<filename file="23.txt" link="inout"/>
<filename file="statfile.tbl" link="input"/>
<filename file="correct.tbl" link="inout"/>
<filename file="c1.fits" link="inout"/>
<filename file="c2.fits" link="inout"/>
<filename file="c3.fits" link="inout"/>
<filename file="add.jpg" link="output"/>
<!-- Part 2: Definition of Jobs -->
<job id="j1" name="mProject">
   <argument>-X -x 0.99366 -i<file name ="1.fits"/> <file
name="big_region.hdr"/>-o <file name= "p1.fits"/></argument>
</job>
<job id="j2" name="mProject">
   <argument>-X -x 0.99376 -i<file name= "2.fits"/> <file
name="big_region.hdr"/>-o <file name= "p2.fits"/></argument>
</job>
<job id="j3" name="mProject">
   <argument>-X -x 0.99330 -i<file name= "3.fits"/> <file
name="big_region.hdr"/>-o<file name= "p3.fits"/></argument>
</job>
<job id="j4" name="mDiffFit">
 <argument>-s -i<file name="p1.fits"/><file name= "p2.fits"/>-o<file name=
"12.txt"/> </argument>
</job>
<job id="j5" name="mDiffFit">
  <argument>-s -i<file name="p2.fits"/> <file name= "p3.fits"/>-o<file
name="23.txt"/> </argument>
</job>
<job id="j6" name="mBgModel">
 <argument>-i 100000 -i<file name= "12.txt"/> <file name= "23.txt"/> <file
name="statfile.tbl"/>-o<file name= "correct.tbl"/> </argument>
</job>
<job id="j7" name="mBackground">
 <argument>-t -i<file name="p1.fits"/> <file name= "correct.tbl"/>-o<file
name= "c1.fits"/></argument>
</job>
<job id="j8" name="mBackground">
 <argument>-t -i<file name="p2.fits"/> <file name= "correct.tbl"/>-o<file
name= "c2.fits"/></argument>
</job>
<job id="j9" name="mBackground">
 <argument>-t -i<file name="p3.fits"/> <file name= "correct.tbl"/>-o<file
name= "c3.fits"/></argument>
</job>
<job id="j10" name="mAdd">
 <argument>-e -i<file name="c1.fits"/> <file name= "c2.fits"/> <file name=
"c3.fits"/>-o<file name="add.jpg"/> </argument>
</job>
 <!-- Part 3: Control-Flow Dependencies -->
<child ref="j4">
 <parent ref="j1"/>
 <parent ref="j2"/>
</child>
<child ref="j5">
 <parent ref="j2"/>
 <parent ref="j3"/>
</child>
<child ref="j6">
 <parent ref="j4"/>
 <parent ref="j5"/>
</child>
<child ref="j7">
 <parent ref="j1"/>
 <parent ref="j6"/>
</child>
<child ref="j8">
 <parent ref="j2"/>
 <parent ref="j6"/>
</child>
<child ref="j9">
 <parent ref="j3"/>
 <parent ref="j6"/>
</child>
<child ref="j10">
 <parent ref="j7"/>
 <parent ref="j8"/>
 <parent ref="j9"/>
</child>
```

In above DAX, the workflow specification in bold is corresponding to the forward slice in Fig. 4 and the specification in black (not bold) is corresponding to the backward slice in Fig. 5. From this DAX, we can see that workflow slice can greatly refine the structure of the original workflow.

RELATED WORKS AND CONCLUSION

The idea of slicing was introduced by Weiser (1984). Since, then, there have been lots of literatures of different slicing techniques, such as dynamic program slicing (Mund *et al*., 2002; Ward and Zedan, 2010), conditioned program slicing (Canfora *et al*., 1998), object-oriented program slicing (Mohapatra *et al*., 2005; Barpanda and Mohapatra, 2011), etc. Up to now, slicing technique has been mainly applied in software engineering. In addition to program slicing, Doctor Zhao (1998) applied the slicing technique to software architecture. Software architecture slicing gives a way to provide knowledge about high-level structure of a software system, thus can be used in architectural understanding, testing, reengineering, maintenance and reuse. Except normal program languages, Silva (2006, 2007) proposed a method to Extensive Makeup Language documents slicing which can be used in web technologies.

This study denotes a concept and the method to scientific workflow slice which give a way for workflow users to find the custom knowledge of a workflow efficiently. Our approach can be seemed as a kind of workflow data filter which is expected to be used in data-intensive scientific workflow systems, such as workflow provenance, fault-tolerance, verification and reuse, etc. How to apply workflow slicing technology in these areas is our future works.

ACKNOWLEDGMENTS

The authors wish to thank Weiwei Chen. He is a member of Pegasus Collaborative Computing Group. With his help, we know more about Pegasus, DAX and Montage workflows. This work is supported by the Science and Technology Development Plan of Shandong Province of China (No. 2011GGH22203).

REFERENCES

Barpanda, S.S. and D.P. Mohapatra, 2011. Dynamic slicing of distributed object-oriented programs. IET Software, 5: 425-433.

Berriman, G.B., E. Deelman, J.C. Good, J.C. Jacob and D.S. Katz *et al.*, 2004. Montage: A grid-enabled engine for delivering custom science-grade mosaics on demand. Proceedings of SPIE Conference on Optimizing Scientific Return for Astronomy through Information Technologies, Volume 5493, June 21, 2004, Glasgow, Scotland.

Binkley, D.W. and K.B. Gallagher, 1996. Program slicing. Adv. Comput., 43: 1-50.

Callaghan, S., E. Deelman, D. Gunter, G. Juve and P. Maechling *et al.*, 2010. Scaling up workflow-based applications. J. Comput. Syst. Sci., 76: 428-446.

Canfora, G., A. Cimitile and A. de Lucia, 1998. Conditioned program slicing. Inform. Software Technol., 40: 595-607.

Deelman, E., D. Gannon, M. Shields and I. Taylor, 2006. Workflows and e-Science: An overview of workflow system features and capabilities. Future Gener. Comput. Syst., 25: 528-540.

Elmroth, E., F. Hernandez and J. Tordsson, 2010. Three fundamental dimensions of scientific workflow interoperability: Model of computation, language and execution environment. Future Gener. Comput. Syst., 26: 245-256.

Goble, C. and D. De Roure, 2009. The Impact of Workflow Tools on Data-Centric Research. In: Data Intensive Computing: The Fourth Paradigm of Scientific Discovery, Hey, T., S. Tansley and K. Tolle (Eds.). Microsoft Research, USA., ISBN-13: 9780982544204, pp: 137.

Katz, D.S., G.B. Berriman and R.G. Mann, 2010. Collaborative Astronomical Image Mosaics. In: Reshaping Research and Development Using Web 2.0-Based Technologies, Baker, M. (Ed.). Nova Science Publishers, USA .

Lim, C., S. Lu, A. Chebotko and F. Fotouhi, 2011. Storing, reasoning and querying OPM-compliant scientific workflow provenance using relational databases. Future Gener. Comput. Syst., 27: 781-789.

McPhillips, T., S. Bowers, D. Zinn and B. Ludascher, 2009. Scientific workflow design for mere mortals. Future Gener. Comput. Syst., 25: 541-551.

Mohapatra, D.P., R. Mall and R. Kumar, 2005. Computing dynamic slices of concurrent object-oriented programs. Inform. Software Technol., 47: 805-817.

Mund, G.B., R. Mall and S. Sarkar, 2002. An efficient dynamic program slicing technique. Inform. Software Technol., 44: 123-132.

Pegasus Workflow Management System, 2013. Pegasus 4.3 user guide: Chapter 4 creating workflows. http://pegasus.isi.edu/wms/docs/latest/creating_workflows.php#abstract_workflows.

Romano, P., 2008. Automation of in-silico data analysis processes through workflow management systems. Briefings Bioinform., 9: 57-68.

Silva, J., 2006. Slicing XML documents. Electron. Notes Theor. Comput. Sci., 157: 187-192.

Silva, J., 2007. A program slicing based method to filter XML/DTD documents. Proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science, January 20-26, 2007, Harrachov, Czech Republic, pp: 771-782.

Ward, M. and H. Zedan, 2010. Combining dynamic and static slicing for analysing assembler. Adv. Comput., 75: 134-175.

Weiser, M., 1984. Program slicing. IEEE Trans. Software Eng., SE-10: 352-357.

Zhao, J.J., 1998. Applying slicing technique to software architectures. Proceedings of the 4th IEEE International Conference on Engineering Complex Computer Systems, August 10-14, 1998, Monterey, CA., USA., pp: 87-99.