Facilitating Reproducible Research By Investigating Computational Metadata

Priyaa Thavasimani*, Paolo Missier*
*School of Computing Science, Newcastle University, UK
Emails:{p.thavasimani2, paolo.missier}@newcastle.ac.uk

Abstract—**Computational workflows involves series of steps in which data is generated, manipulated, analysed and transformed. Researchers uses tools and techniques to capture the provenance associated with the data to aid reproducibility. The metadata collected not only helps in reproducing the computation but also aids in comparing the original and reproduced computations. In this paper, we present an approach "Why-Diff" to analyse the difference between two related computations by changing the artifacts and how the existing tools YesWorkflow and NoWorkflow records the changed artifacts.**

I. Introduction

Scientific research progresses when discoveries are reproduced and verified. This research emphasises the reproducibility of computations rather than simple repeatability or replicability. Execution of reliable Reproducible research requires more than just good tools. Most computations that are reproduced do not guarantee the correctness of results because of the underlying fact that they use different artifacts. Computations are redone because of two primary reasons. The first is to validate the results and the second reason is to enable reuse of the results by other researchers, who may then be able to extend or modify the experimental method. The term 'reproducibility' is often used in research of redoing scientific experiments. Researchers give a semantic distinction between replicability and reproducibility. Replicability is redoing experiment in exactly the same way, while reproducibility focuses on the result being verified rather than on the specific method used to achieve the result. The motivation of reproduction is on concepts rather than artifacts to recreate the essence of the experiment rather than the experiment itself. Mere redoing of existing computations has no claim for generalisability. The reproducibility is ambitious as it promotes meta studies, where the ideas of multiple independent studies are combined and produce better results. Though there are existing technologies that makes Reproducible Research possible, improper use of computational tools and software can lead to spectacularly incorrect results. Guaranteeing Reproducible Research requires a system that is rigorous in keeping track of research activities. There are existing provenance capturing tools of which some records prospective provenance while some others records retrospective provenance. The main aim of the research is to investigate and exploit the metadata of computational experiments, inorder to help researchers understand changes between outcomes of related computational experiments over time.

II. Background and Related Work

Redoing computations often provoke a question whether it aims at repeatability, replicability or reproducibility. Reproduction and Replication of reported scientific results is a widely discussed topic within the scientific community [1][2][3][4][5][6][7][8]. This work also emphasizes the importance of reproducibility saying it uncovers the mistakes (or fraud) more readily than replication which provides a baseline and facilitates extending and building on previous work. A classification was given by C. Drummond [7] saying replicability to denote situations where the previous experiment is redone in exactly the same way, whereas in reproducibility the focus is on the result being verified and not on the method to achieve this result. The motivation of reproduction is on concepts rather than artifacts we try to recreate the essence of the experiment rather than the experiment itself [8]. In this project, we address the reproducibility of computations, as we do not aim to achieve exact incarnation of the existing computation, but rather varying, augmenting and remodeling the existing methods and also by applying parameter sweeps.

Reproducibility of computations is a challenging task because of the underlying fact that it involves change at multiple levels, some of which are not within the control of the experimenter [8]. Long-term reproducibility is hard to achieve due to the factors that include the use of specialized hardware, proprietary data and inevitable changes in the hardware and software environments. Freire et al. [9]

have introduced three criteria to characterize the level of reproducibility of computations. The three criteria include Depth which evinces how much of an experiment is made available, Portability indicates whether the experiment can be reproduced on the original environment, similar or on a different environment, Coverage that specifies how much of the experiment can be reproduced. The Coverage can either be full or partial. In general, computations are viewed as a black box which takes input data and produce output data. In order to reproduce the computational experiments, lot more details are needed to be considered other than just the input and output. Workflows are widely used to represent and execute computational experiments. Capturing and exploiting provenance information is considered to be important in case of computational reproducibility [10][11][12][13]. Tools such as Git, CVS and SVS help in recording and tracking of program files that changes over time. Tools that record provenance of computations include E-Science Central [14], Reprozip [15], Pegasus [16], Galaxy, VisTrails, Kepler, Taverna, Madagascar, etc [17]. Workflow Management System like Taverna [18] and Kepler [19] facilitates the user to design workflows using graphical editor. After capturing the provenance traces from the available tools, automating the computation is another milestone. Available virtualisation technology such as Virtual Machines containers like Docker are able to persistently capture a computational environment, so that they can be deployed in the cloud. Projects like RunMyCode.org and recomputation.org exploit this kind of technology, facilitating construction and availability of computational environments in the cloud. TOSCA is a standard by OASIS that aids in automatic deployment [20] by providing a technology-independent, abstract and extensible model to describe workflows. The studies that are listed above facilitates replicability which forms the basis of reproducibility. Reproducibility implies atleast some changes have been introduced in the experiment, thus exposing different features. The result of comparison of two provenance traces can give surprising or unexpected results even though we have reproduced the same experiment with minor changes and further gives rise to questions, How this data was generated? Reasoning about the sequence of steps that led to a particular result is the real goal of reproducing anothers work, and tools to manipulate the provenance of results are key enabler [21]. In addition to reproducing computations, it has become essential to analyse and monitor data transformations. Provenance tracking is generally classified into two types - Backward and Forward. Backward tracking is finding the input subsets that contributed to a given output element whereas Forward Provenance is determining which output elements were derived from a particular input element. Researchers classify Provenance captured by the system into two basic types [22]: Prospective and Retrospective. Prospective provenance captures the abstract specification of the computational tasks like workflow procedure calls and data dependencies whereas retrospective provenance captures more detailed information like execution environment, recordings of where and when each procedure was run and how it executed. In addition [23], the author classified the Provenance for single transformations as Map, Reduce, Union and Split Provenance. Workflow Management Systems not only capture the provenance, but also capture the workflow evolution and offers all the data for users to analyze it. Despite of rich provenance captured by Workflow Management Systems, a vast number of computational workflows are being developed using general purpose scripting languages such as Python, R, and Matlab. YesWorkflow and noWorkflow are the tools that captures the provenance of independent scripts. YesWorkflow [24] extracts the latent information from scripts, exports, visualize in graph form. Given a script, YesWorkflow generates the workflow based on user annotations which generally captures prospective provenance. noWorkflow [25] (not only Workflow) system uses Python runtime profiling functions to generate provenance traces that reflect the processing history of the script. noWorkflow captures three types of provenance - definition, deployment, and execution provenance.

III. WHY-DIFF APPROACH

In broader perspective, the dependencies of computational experiments include elements at all levels of the architectural stack, namely: Hardware configuration, OS version and configuration, Software libraries, external services and external data sources (which are often accessed through web-based services).
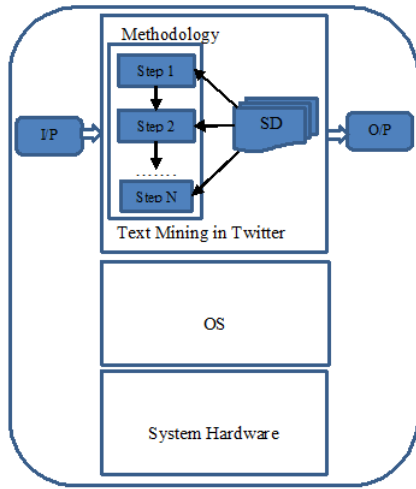
Fig. 1: Architecture stack with dependencies at multiple levels.

Some of these dependencies are easily satisfied by simple virtualisation of the computing environment. For instance, in the example stated above, Hardware and OS requirements can be met just by allocating a VM with the correct OS version, on a cloud infrastructure that can satisfy the HW requirements. Software dependencies, on the other hand, may be more challenging. Reproducing this scenario requires tracking the changes in some of these dependencies over time. Computations are reproduced to achieve better results. Reproducing a computation can give unexpected results as it involves different artefacts. As depicted in the Figure 1, a computation has dependencies at 3 levels such as Hardware, OS and Software level. As hardware and OS level does not impact the computational results much, we skipped doing hardware and OS level comparisons. In the architectural stack, the SD represents Software Dependencies. Rather than just replicating a computation, it is about changing input or any other dependencies (without breaking the setup execution environment) and assessing the effect on the outcome. This allows the other researchers to mix and match their own methods with the existing system to achieve better results. Extending the idea of reproducibility is the conceptual replication, in which the essential conclusions of a study are examined, intentionally using different methods than in the original study. Researchers try to extend previous work with changing artefacts. The project aim to consider 4 possible cases for reproducing computations. We have used P, X, D, Y to refer to Program, Input, Dependency and Output of the original computation,

respectively. Similarly, we denote the same quantities in the reproduced computation by P', X', D' and Y', respectively.

In "Why-Diff", we consider four possible cases for reproducing computations, as follows:

1) $\{P, X, D'\}$ - Changing Dependency keeping Program, input unchanged.
2) $\{P, X', D\}$ - Changing Input keeping Program, Dependency unchanged
3) $\{P', X, D\}$ - Changing Program keeping Dependency, input unchanged
4) $\{P', X', D'\}$ - All artefacts changed

Here, we tried to apply first case in which we changed the Dependency keeping Program and Input as constant while redoing computation. The provenance collected from the original computation will vary from Coarse-grained to fine-grained details. We classify the provenance-level based on the provenance details available. The classifications are as follows:

1) Full-Provenance: This is a best-case which contains fine-grained provenance details including definition, deployment and execution provenance.

2) Partial-Provenance: This is an average-case which includes only the definition and deployment provenance but not the execution provenance. In other words, this case contains only prospective provenance but not the retrospective provenance.

3) Mandatory-Provenance: This is a real-world scenario which includes only the definition provenance.

Having this provenance details, we aim to answer why the outcome from 2 related computations are different.

IV. MODEL USECASE FOR COMPUTATIONAL REPRODUCIBILITY

Sentiment analysis or opinion mining has emerged as an active domain among the research fraternity because enormous amount of heterogeneous data is continuously increasing every day by the users via

www, viz., e-commerce websites, social networks, discussion forums, blogs etc. Motivated by the Natural Language Processing techniques, We have developed a usecase Twitter Sentiment Analysis which is a python script which takes in static tweets and visualise the sentiment score percentage Positive, Negative and Neutral in a pie chart. Lot of Natural Language Processing tools are available to process the human language to classify whether it is subjective or objective and orient its polarity as positive, negative and neutral. TextBlob and NLTK are similar Natural Language Processing tools which provides API which is used to analyse the tweets and compute sentiment score for the text. It helps in converting an unstructured text to structured text which is easily manageable. While reproducing the usecase, we have intensionally used NLTK (Refer Fig 3) replacing TextBLOB (Refer Fig 2), for the same set of tweets. Positive, Negative and Neutral sentiment percentages using TextBLOB are 38.0%, 12.0% and 50.0% respectively. Whereas NLTK computes 18.0%, 17.0% and 65.0% as Positive, Negative and Neutral percentages respectively. In this case, $Y \neq Y'$ when trying to mock the case $\{P, X, D'\}$ where D refers TextBLOB and D' refers NLTK respectively.
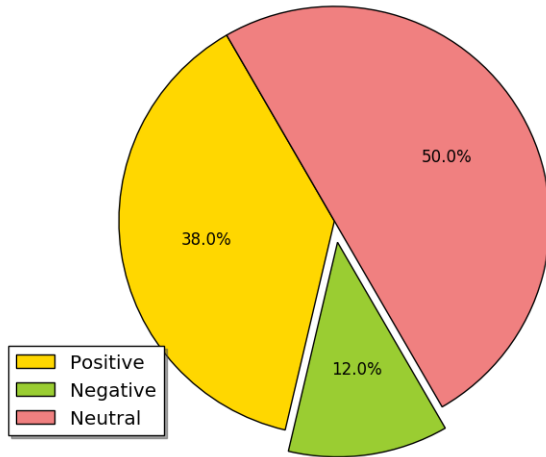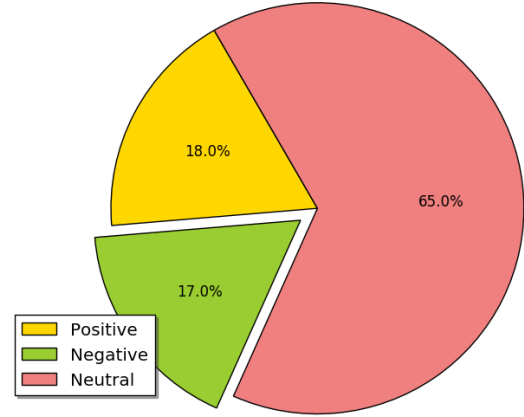

Fig. 3: Sentiment Scores using NLTK

We have observed a striking difference in the results when using different software dependencies keeping the Program and Input unchanged. We viewed TextBLOB and NLTK as a blackbox and recorded the provenance of the two scripts without bothering about the algorithm design of TextBLOB and NLTK. Motivated by the different results from related computations, we have analysed the provenance data collected by YesWorkflow and NoWorkflow.

V. YESWORKFLOW

YesWorkflow is one of the approach to capture prospective provenance from scripts. It helps to interpret the user annotations and visualise the graph that shows the computational steps and data flow in the script.We used YesWorkflow to visualize the graph to extract the API dependency (TextBLOB or NLTK) and retrieved two graphs showing the dependency which is depicted in Fig 4 and Fig 5 respectively.
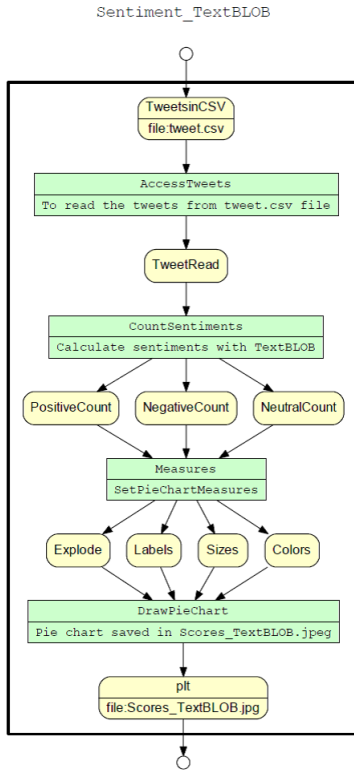

Fig. 2: Sentiment Scores using TextBLOB

Fig. 4: TextBLOB Visualised by YesWorkflow
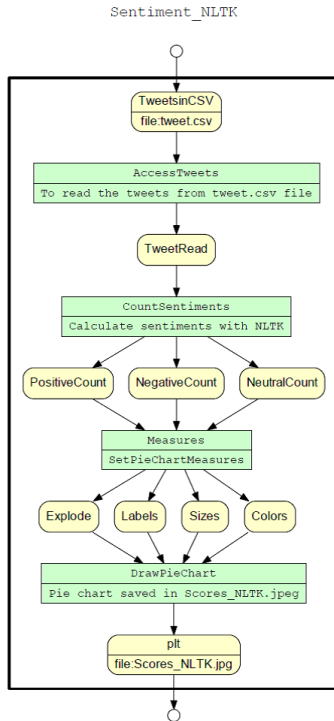

Fig. 5: NLTK Visualised by YesWorkflow

In addition to the visualization, the YesWorkflow

facilitates querying the workflow steps. The Following Prolog query is used to extract the dependent API.

```
banner( 'YW_Q4',
        'Describe CountSentiments step?',
        'yw_q4(StepName,Description)').
[user].
:- table yw_q4/2.
yw_q4(StepName, Description) :-
    yw_workflow_script(WorkflowId,_,_,_),
    yw_workflow_step(StepId, 'CountSentiments',
WorkflowId, _, _, _),
    yw_description(program, StepId, StepName,
Description).
end_of_file.
printall(yw_q4(_,_)).
```

Fig. 6: YesWorkflow Query

The Fig. 7 and Fig. 8 depicts the query result for extracting the dependency of TextBLOB and NLTK respectively.


```
YW_Q4 : What workflow steps comprise the
top-level workflow?

yw_q4(StepName,Description)

yw_q4('CountSentiments','Calculate
sentiments with TextBLOB).
```

Fig. 7: TextBLOB Queried by YesWorkflow


```
YW_Q4 : What workflow steps comprise the
top-level workflow?

yw_q4(StepName,Description)

yw_q4('CountSentiments','Calculate
sentiments with NLTK').
```

Fig. 8: NLTK Queried by YesWorkflow

VI. NOWORKFLOW

Noworkflow is another tool that captures the retrospective provenance automatically. It does not require user annotations for recording provenance. NoWorkflow defines one execution of the experiment as Trial. Here, the execution of Sentiment analysis with TextBLOB is referred as Trial 1 whereas the execution with NLTK referred as Trial 2.
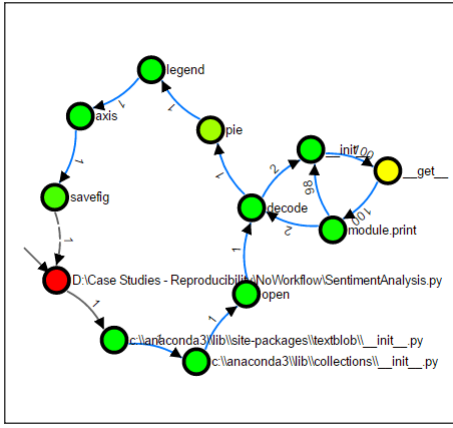
Fig. 9: TextBLOB Visualised by NoWorkflow

The Fig.9 refers provenance graph of TextBLOB execution whereas Fig.10 refers provenance graph of NLTK execution.
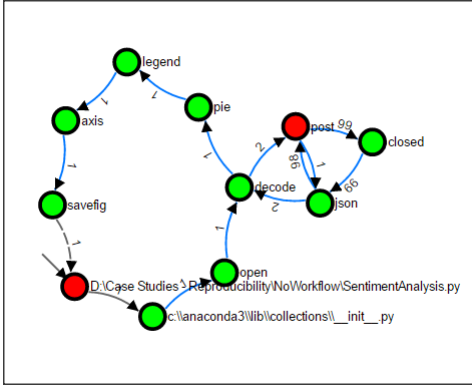

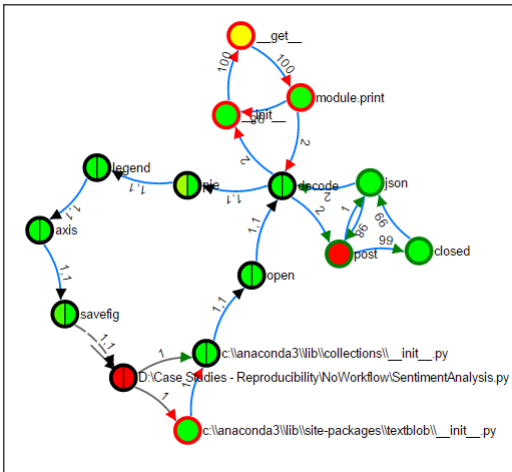Fig. 10: NLTK Visualised by NoWorkflow

Fig. 11: Diff TextBLOB and NLTK Visualised by NoWorkflow

NoWorkflow has a command called Diff which

yields a difference graph between 2 executions. The difference between TextBLOB and NLTK is depicted in the Fig11. As you can see, the Trial 1 uses the module TextBlob whereas Trial 2 does not.

VII. CONCLUSION AND FUTURE WORK

We have shown as well queried the provenance captured by YesWorkflow and NoWorkflow by changing the dependency and keeping the program and input constant (i.e. $\{ P, X, D' \}$). We intend to vary the program and inputs and to analyse the difference for the variation. In addition to these high-level changes, we are aiming to vary the variables, function parameters and return values and to investigate the meta-data captured. It would also be interesting to code the same usecase in different languages like Python, R, Matlab and to visualise and query them to know if and why there is a difference.

VIII. ACKNOWLEDGEMENT

REFERENCES

[1] N. Barnes, "Publish your computer code: it is good enough," *Nature*, vol. 467, no. 7317, pp. 753–753, 2010. [Online]. Available: http://www.nature.com/doifinder/10.1038/467753a

[2] M. Schwab, N. Karrenbach, and J. Claerbout, "Making scientific computations reproducible," *Computing in Science & Engineering*, vol. 2, no. 6, pp. 61–67, 2000. [Online]. Available: http://ieeexplore.ieee.org.ezp-prod1.hul.harvard.edu/ielx5/5992/19077/00881708.pdf?tp=&arnumber=881708&isnumb

[3] J. P. Mesirov, "Accessible Reproducible Research," vol. 327, no. January, pp. 415–417, 2010.

[4] A. Morin, J. Urban, P. D. Adams, I. Foster, A. Sali, D. Baker, and P. Sliz, "Research priorities. Shining light into black boxes." *Science (New York, N.Y.)*, vol. 336, no. 6078, pp. 159–60, 2012. [Online]. Available: https://www.rdcep.org/sites/default/files/publications/Science-2012-Morin-159-60.pdf\nhttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=4203337

[5] G. Wilson, D. A. Aruliah, C. T. Brown, N. P. Chue Hong, M. Davis, R. T. Guy, S. H. D. Haddock, K. D. Huff, I. M. Mitchell, M. D. Plumbley, B. Waugh, E. P. White, and P. Wilson, "Best Practices for Scientific Computing," *PLoS Biology*, vol. 12, no. 1, 2014.

[6] J. Vitek and T. Kalibera, "Repeatability, Reproducibility, and Rigor in Systems Research," *Proceedings of the ninth ACM international conference on Embedded software - EMSOFT '11*, pp. 33–38, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2038642.2038650

[7] D. C. Drummond, "Replicability is not reproducibility: Nor is it good science," *Proceedings of the Evaluation Methods for Machine Learning Workshop 26th International Conference for Machine Learning*, no. 2005, pp. 1–4, 2009.

[8] D. G. Feitelson, "From repeatability to reproducibility and corroboration," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 3–11, 2015. [Online]. Available: http://dl.acm.org/citation.cfm?id=2723875

[9] J. Freire, P. Bonnet, and D. Shasha, "Computational reproducibility: state-of-the-art, challenges, and database research opportunities," *Proceedings of the 2012 ACM SIGMOD*, pp. 593–596, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2213908

[10] S. Miles, S. C. Wong, W. Fang, P. Groth, K. P. Zauner, and L. Moreau, "Provenance-based validation of e-science experiments," *Web Semantics*, vol. 5, no. 1, pp. 28–38, 2007.

[11] S. Davidson and J. Freire, "Provenance and scientific workflows: challenges and opportunities," *Proceedings of the 2008 ACM SIGMOD . . .*, pp. 1–6, 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1376772

[12] G. T. Lakshmanan, F. Curbera, J. Freire, and A. Sheth, "Provenance in Web Applications," *IEEE Internet Computing*, vol. 15, no. 1, pp. 17–21, 2011. [Online]. Available: http://www.computer.org/portal/web/csdl/magazines/internet;jsessionid=d2017d11

[13] P. Groth, S. Miles, S. Modgil, N. Oren, M. Luck, and Y. Gil, "Determining the trustworthiness of new electronic contracts," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5881 LNAI, pp. 132–147, 2009.

[14] H. Hiden, S. Woodman, P. Watson, and J. Caa, "Developing cloud applications using the e-Science Central platform," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 371, no. 1983, jan 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/23230161

[15] F. Chirigati, D. Shasha, and J. Freire, "{ReproZip}: {Using} {Provenance} to {Support} {Computational} {Reproducibility}," *Proceedings of the 5th {USENIX} {Workshop} on the {Theory} and {Practice} of {Provenance}*, pp. 1:1—-1:4, 2013. [Online]. Available: http://dl.acm.org.proxy.lib.chalmers.se/citation.cfm?id=2482949.2482951

[16] J. Kim, E. Deelman, Y. Gil, G. Mehta, V. Ratnakar, and M. Rey, "Provenance Trails in the Wings / Pegasus System Wings / Pegasus : Creating and Executing Large Workflows," *Information Sciences*, vol. 20, no. 5, pp. 1–11, 2007. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.121.2529

[17] V. Stodden, J. Borwein, and D. Bailey, "Setting the default to reproducible," *Computational Science Research. SIAM News*, vol. 46, pp. 4–6, 2013.

[18] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, "Taverna: A tool for building and running workflows of services," *Nucleic Acids Research*, vol. 34, no. WEB. SERV. ISS., pp. 729–732, 2006.

[19] B. Ludäscher, C. Berkley, M. Jones, and E. A. Lee, "Scientific Workflow Management and the Kepler System ," *Electrical Engineering*, vol. 0078296, pp. 1–19, 2005.

[20] R. Qasha, J. Cala, and P. Watson, "Towards Automated Workflow Deployment in the Cloud Using TOSCA," in *2015 IEEE 8th International Conference on Cloud Computing*, New York, 2015, pp. 1037–1040. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7214157

[21] K.-K. Muniswamy-Reddy, P. Macko, and M. Seltzer, "Provenance for the Cloud," *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, pp. 14–15, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855511.1855526\nhttp://www.usenix.org/event/fast1 reddy.pdf

[22] D. Koop, E. Santos, B. Bauer, M. Troyer, J. Freire, and C. T. Silva, "Bridging Workflow and Data Provenance Using Strong Links," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010, vol. 6187 LNCS, pp. 397–415. [Online]. Available: http://link.springer.com/10.1007/978-3-642-13818-8_28

[23] R. Ikeda, H. Park, and J. Widom, "Provenance for generalized map and reduce workflows," *Proceedings of the Fifth CIDR*, pp. 273–283, 2011. [Online]. Available: http://ilpubs.stanford.edu:8090/985/

[24] T. McPhillips, T. Song, T. Kolisnik, S. Aulenbach, K. Belhajjame, K. Bocinsky, Y. Cao, F. Chirigati, S. Dey, J. Freire, D. Huntzinger, C. Jones, D. Koop, P. Missier, M. Schildhauer, C. Schwalm, Y. Wei, J. Cheney, M. Bieda, B. Ludaescher, and B. Lud, "YesWorkflow: A User-Oriented, Language-Independent Tool for Recovering Workflow Information from Scripts," *International Journal of Digital Curation*, vol. 10, no. 1, pp. 1–15, 2015. [Online]. Available: http://arxiv.org/abs/1502.02403

[25] "noWorkflow: Capturing and Analyzing Provenance of Scripts."