Supporting Sustainable Process Documentation

Markus Gärtner[1](✉), Uli Hahn[2], and Sibylle Hermann[3]

[1] Institute for Natural Language Processing,
University of Stuttgart, Stuttgart, Germany
markus.gaertner@ims.uni-stuttgart.de

[2] Communication and Information Centre, Ulm University, Ulm, Germany
uli.hahn@uni-ulm.de

[3] University Library, University of Stuttgart, Stuttgart, Germany
sibylle.hermann@ub.uni-stuttgart.de

Abstract. In this paper we introduce a software design to greatly simplify the elicitation and management of process metadata for researchers. Detailed documentation of a research process not only aids in achieving reproducibility, but also increases usefulness of the documented work for others as a cornerstone of good scientific practice. However, in reality, time pressure together with the lack of simple documentation methods makes documenting workflows an arduous and often neglected task. Our method for a clean process documentation combines benefits of version control with integration into existing institutional infrastructure and a novel schema for describing process metadata.

1 Introduction

Lately the term "reproducibility crisis" has frequently been brought up in the context of criticism on research quality in general. However, the German Research Foundation (DFG) correctly emphasizes[1] that reproducibility[2] is only one of many quality measures for good research. Furthermore, it asks for increased attention to questions raised in the close-by area of research data management (RDM). The topic is also in line with recent survey results[3] which stress the insecurity of many researchers when it comes to RDM and the related issues of research data sustainability.

Following the idea of Claerbout and Karrenbach (1992) that articles are only advertisement for the actual scholarship, due diligence should also be exercised in compiling and publishing documentations of research processes alongside the results. This helps especially for tasks that naturally can be reproducible if documented thoroughly, such as computations with strictly deterministic outcomes.

[1] Press release by the German Research Foundation (DFG) on "Replicability of Research Results", April 25th 2017.

[2] We do not intend to delve into the deeper semantic discussion of "reproducibility" vs. "replicability" and treat both terms as synonyms here.

[3] By project bwFDCommunities http://bwfdm.scc.kit.edu or Humboldt-University Berlin http://nbn-resolving.org/urn:nbn:de:kobv:11-100213001.

G. Rehm and T. Declerck (Eds.): GSCL 2017, LNAI 10713, pp. 284–291, 2018.
https://doi.org/10.1007/978-3-319-73706-5_24

But even if a process conceptually defies reproduction (e.g., results of strictly qualitative analysis) a good documentation can increase usability.

Today's research practice however is dominated by competitive and time pressures. Together with the focus on producing publishable end results combined with lack of gratification for detailed process documentation this leads to a serious neglect of documentation efforts. Our project aims at filling that gap with the software tool described in this paper. The goal is to support scientists in creating process documentation already during the workflow with minimal effort. This concept is being exemplified with use cases from researchers in computational linguistics and digital humanities where a plethora of multifaceted workflows exist.

We contextualize our work in Sect. 2 and introduce our schema for process metadata in Sect. 3. Section 4 provides an overview of our software design and shows possibilities of interconnections to related infrastructure. In Sect. 5 we summarize the findings and hint at possibilities of how this concept can be integrated within a broader context.

2 Related Work

In the context of documenting a research process, systems for two slightly distinct fields are relevant: Workflow management and workflow tracking.

A *Workflow management system* (WMS) is software that helps in setting up workflows as collections of interdependent (executable) steps. The list of (commercial) WMSs for general-purpose or enterprise use is extensive, but their usability for specialized research workflows is limited. For certain research fields dedicated WMS instances have emerged. *GenePattern*[4] (Reich et al. 2006) for instance allows the creation, management and execution of analysis pipelines for genome research.

The task of *workflow tracking* is more reactive and involves documenting steps during or after their execution. Tools similar to *YesWorkflow* (McPhillips et al. 2015) offer the ability to annotate data flows and operations on the code level and to receive a graph visualization of the implicit workflow. They are however primarily aimed at script-heavy workflows. More related to software development, version control systems like Git[5], Apache Subversion (SVN)[6] or others provide sufficient functionality for tracking and documenting complex collaborative development workflows.

In stark contrast to the elaborate solutions listed so far, in practice it is not uncommon for researchers to document their workflow simply by means of a local Word or Excel file. Reasons for this given in surveys include the complexity of many workflow management or tracking systems, the barrier of entry for technically unskilled users and available systems essentially being not specialized enough for a given workflow. In addition many solutions focus on

[4] http://www.genepattern.org.
[5] https://git-scm.com.
[6] https://subversion.apache.org.

creation and maintenance of executable analysis or processing pipelines, making them unsuited for workflows that involve manual steps, such as annotation or curation.

By splitting process documentation into metadata for *resources* (objects) used in it and the *actions* performed, the need to interface with existing metadata infrastructure becomes evident. Initiatives like CLARIN (Hinrichs and Krauwer 2014) already provide wide coverage of metadata repositories for communities in computational linguistics and digital humanities. While having rich object metadata available is by no means a sufficient replacement of proper process documentation, it does provide a valuable foundation to build on.

3 Process Metadata

We follow the more common convention of modeling workflows as directed acyclic graphs (DAG) (Deelman et al. 2009) where each node represents a single step. As pointed out in Sect. 2, there are already established initiatives and systems for the provision or archiving of object metadata. That is metadata associated with an individual resource, usually created and archived after reaching a mature state in the resource's development or creation lifecycle. Rich in detail, such metadata records are also typically following schemas specific to the field of research they originated from. For example in the context of CLARIN the Component MetaData Infrastructure (CMDI)[7] is used for entries in the CLARIN Virtual Language Observatory (VLO)[8]. Due to such infrastructures also providing means for persistent identification of individual records[9], the metadata schema in this section is focused solely on the process itself. To this end our schema defines the following fields for a single workflow step (with the field's multiplicity in round brackets if it is optional):

Title. User-defined short label for the workflow step. This is meant to be very compact and can also be reduced to simply state the type of the task performed, such as "Annotation".

Description. This more detailed description of the workflow step is meant to be a human readable explanation that should contain enough information for another researcher to understand what was done. It is also the place where basically arbitrary additional notes can be placed. It will help to find and keep track of decisions or expectations and to raise the reusability for others.

Input (0..n). Resources used to perform the action. This includes an extremely diverse spectrum of resources which are in turn highly task-specific. They can range from local resources of arbitrary type (corpora, model files, configuration files, pictures, audio, etc.) to online resources (e.g., public annotation guidelines) to "pointers" at real objects such as books that don't exist in digitized form.

[7] http://www.clarin.eu/cmdi.
[8] https://vlo.clarin.eu.
[9] E.g., metadata for the TIGER Corpus: http://hdl.handle.net/11022/1007-0000-0000-8E2D-F.

Output (0..n). Resources generated by the action. Unlike input resources, these usually represent exclusively local files, since the workflow is assumed to take place on a local system.

Tool (0..1). The executable resource used for processing. This is either a local program or a web-service and in both cases command line parameters or settings can and should be recorded.

Person (0..n). Human subjects involved in the workflow step. Similar to `Input`, the content of this field is quite diverse, including, but not limited to, annotators, curators or experiment participants.

Custom properties (0..n). Arbitrary classic textual key-value metadata entries that can be added, primarily to provide machine readable storage options for metadata created by future plugins or if the user wishes to include structured information beyond the free-text in the `Description`.

Complex fields (`Input`, `Output`, `Person`, `Tool`) get assigned one or more typed identifiers. These identifiers take the form of <`type, id`> where `id` is the actual identification string such as a name or web address and `type` defines how that `id` is to be interpreted. These identifiers can also be used to link resources or persons to public repositories or databases (e.g., VLO-Handles for CLARIN resources or an ORCID[10] for registered researchers).

The standard serialization format for the process metadata is JSON[11]. Our tool uses it to store process metadata locally as described in Sect. 4.1 and for exporting (parts of a) workflow documentation. Since JSON is a very simple and also widely used format, it is easily possible to convert the output, or process it with other systems.

4 Architecture

In this section we provide an overview of our architecture, especially the design of the client software which is being developed in our project. The core component is a Java application which bundles the user interactions required for tracking and documenting the workflow. Behind the scenes this client uses a local Git repository for tracking changes to file resources in *workspace* folders designated by the user. The client is also meant to provide a wide range of additional functionality, among others the ability to interact with existing data and/or metadata repositories to store or retrieve resources or metadata. We describe some of these features in the following Sects. 4.1 through 4.3.

4.1 Git and Process Documentation

The Git system allows version control of a local folder's content, that is, it tracks changes to files inside this folder so that individual states or "versions" can be recorded and referenced.

[10] https://orcid.org.
[11] http://www.json.org.

Storage of Process Metadata. The recording of a workspace's state is triggered by so called Git "commit" operations. In our workflow model every workflow step corresponds to a single Git commit. Each commit is also accompanied by its respective "commit message". Those messages commonly are human-readable descriptions made by the user to explain the nature or reason of performed modifications. However, our application design completely hides the direct Git interface from the end users as to not overwhelm them with unneeded complexity or functions. This way we can use commit messages to store process metadata (cf. Sect. 3) and thereby directly associate physical changes to the data with matching (formal) documentation.

Unfortunately Git can only automatically detect those local files that represent the "output" of a workflow step (since those have been modified or are new). This means that the completeness of a resulting workflow documentation ultimately still relies on the user. We plan assistive functions in the client that try to suggest involved input resources when recording a workflow step to reduce the effort required by the user. Their implementation is at this point however still an open issue and the correctness and usability will have to be evaluated together with the user community at a later stage.

Increasing Documentation Consistency. Manually documenting a workflow is prone to common mistakes, such as forgetting to include changes to a resource in the documentation or introducing inconsistencies in the descriptions. As stated above, Git cannot guarantee completeness when recording all the input resources used in a workflow step. It does on the other hand track reliably all the changes made to files that are under version control. As a result it makes it impossible to miss modification on tracked files and reminds the user to document all of them. Having the entire workflow history available when recording a new workflow step, also enables the client to detect inconsistent documentation, for instance when the user tries to assign a different description to a resource which has been previously used in another step.

Trying Alternatives. Only very few research workflows ever result in a strictly linear concatenation of performed steps. Usually there are situations during a workflow where an assumption was found to be untrue or an evaluation yielded unsatisfactory results. In those cases the researcher typically "goes back" and pursues an alternative approach by performing a different workflow step on the same previous workspace state (e.g., testing new settings for an analysis tool or training systems on different data, etc.). A workflow graph displays this behavior as branches originating from the same node to concurrent new child nodes. Git offers a similarly named functionality where a workflow can split into independent branches and the folder under version control can be changed to reflect any previously recorded state at will.

Backup and Cooperation. Git represents a decentralized or distributed version control system. While every local repository itself allows full version control, one can also import or export changes from and to a remote repository. In the context of our client this allows the "local workflow" to be connected to a remote Git repository (e.g., an institute or university wide GitLab[12] installation). Benefits of this include for example an additional layer of backup for valuable resources used in the workflow or the ability to cooperatively work on the same "shared" workspace from different locations.

While building upon the Git system offers many advantages, there are also limitations and issues to address when using it for workflow documentation, especially in very resource-heavy computational projects. Since Git basically has to duplicate every file that is kept under version control, this leads to a very high storage consumption when used for already big resources, such as web corpora. As a solution the user can exclude files from being tracked, so that they won't affect performance.

4.2 Client Customizability

With the process metadata outlined in Sect. 3 we can model a very broad spectrum of diverse workflows and therefore make the client usable for researchers in many fields. Different research fields and also universities or institutes often already have individual infrastructures for archiving or management of resources and metadata in place. To not create redundancies the following principles are taken into account for the client design:

Independence. In the most basic version our client requires absolutely no external infrastructure or third-party software to work with, besides the libraries it is shipped with. It will provide the full functionality of workflow documentation and also enable the user to create and store object metadata locally in a simple (customizable) schema following Dublin Core (Powell et al. 2005). In this configuration the user can work completely network-independent and also is not reliant on other infrastructure, making the client very light-weight.

Extensibility. To be able to incorporate the client into existing institutional infrastructure we use a plugin-architecture. This allows for example customized implementations for interfacing with additional repositories to be added.

4.3 External Repositories

In addition to workflow documentation in private domains (Treloar et al. 2007), the client also gives the possibility to collaborate in the shared (but not publicly open) domain, and to publish partial or final results in the public domain. To meet both requirements there are two systems with their respective interfaces that will be supported: For publishing within the shared domain, ResourceSpace[13] is being used. The repository software DSpace[14] (Smith 2002) is used

[12] https://gitlab.com.
[13] https://www.resourcespace.com.
[14] http://www.dspace.org.

for publishing data with a permanent identifier (DOI) in the public domain. DSpace is a popular software for institutional publication repositories. We plan to interface with ResourceSpace for the shared domain, as it offers a better rights management as well as the possibility to share data within defined communities.

5 Outlook

In this paper we introduced our design of a software supporting process documentation. We have shown the essential benefits of using version control software such as Git as a foundation for workflow tracking with a main focus on computational linguistics and digital humanities. In addition, we propose a simple yet very expressive metadata schema to describe individual steps in a research workflow. Keeping those principles separated – namely the distinction between metadata describing *objects* used in a workflow and the *actions* performed – enables the software to be very flexible. As a result it will be fairly easy to adopt the tool to specific needs (of other disciplines) and also to integrate it into the diverse landscape of existing infrastructures.

Acknowledgments. This work was funded by the Ministry for Science, Research and the Arts in Baden-Württemberg (MWK) via the E-Science project "RePlay-DH".

References

Claerbout, J., Karrenbach, M.: Electronic documents give reproducible research a new meaning. In: Proceedings of 62nd Annual International Meeting of the Society of Exploration Geophysics, pp. 601–604 (1992)

Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: an overview of workflow system features and capabilities. Future Gener. Comput. Syst. **25**(5), 528–540 (2009). https://doi.org/10.1016/j.future.2008.06.012. ISSN 0167–739X

Hinrichs, E., Krauwer, S.: The CLARIN research infrastructure: resources and tools for e-humanities scholars. In: Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014, pp. 1525–1531, May 2014. URL http://dspace.library.uu.nl/handle/1874/307981

McPhillips, T.M., Song, T., Kolisnik, T., Aulenbach, S., Belhajjame, K., Bocinsky, K., Cao, Y., Chirigati, F., Dey, S.C., Freire, J., Huntzinger, D.N., Jones, C., Koop, D., Missier, P., Schildhauer, M., Schwalm, C.R., Wei, Y., Cheney, J., Bieda, M., Ludäscher, B.: YesWorkflow: a user-oriented, language-independent tool for recovering workflow information from scripts. CoRR, abs/1502.02403, 2015. URL http://arxiv.org/abs/1502.02403

Powell, A., Nilsson, M., Naeve, A., Johnston, P.: Dublin core metadata initiative - abstract model (2005). White Paper, http://dublincore.org/documents/abstract-model

Reich, M., Liefeld, T., Gould, J., Lerner, J., Tamayo, P., Mesirov, J.P.: GenePattern 2.0. Nat. Genet. **38**(5), 500–501 (2006). https://doi.org/10.1038/ng0506-500. ISSN 1061–4036

Smith, M.: Dspace: an institutional repository from the mit libraries and hewlett packard laboratories. In: Agosti, M., Thanos, C. (eds.) ECDL 2002. LNCS, vol. 2458, pp. 543–549. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45747-X_40

Treloar, A., Groenewegen, D., Harboe-Ree, C.: The data curation continuum: managing data objects in institutional repositories, vol. 13, no. 9/10. D-Lib Magazine, September/October 2007. https://doi.org/10.1045/september2007-treloar. ISSN 1082–9873