IJDC | *General Article*

Incorporating Software Curation into Research Data Management Services

Fernando Rios
University of Arizona

Abstract

Many large research universities provide research data management (RDM) support services for researchers. These may include support for data management planning, best practices (e.g., organization, support, and storage), archiving, sharing, and publication. However, these data-focused services may under-emphasize the importance of the software that is created to analyse said data. This is problematic for several reasons. First, because software is an integral part of research across all disciplines, it undermines the ability of said research to be understood, verified, and reused by others (and perhaps even the researcher themselves). Second, it may result in less visibility and credit for those involved in creating the software. A third reason is related to stewardship: if there is no clear process for how, when, and where the software associated with research can be accessed and who will be responsible for maintaining such access, important details of the research may be lost over time.

This article presents the process by which the RDM services unit of a large research university addressed the lack of emphasis on software and source code in their existing service offerings. The greatest challenges were related to the need to incorporate software into existing data-oriented service workflows while minimizing additional resources required, and the nascent state of software curation and archiving in a data management context. The problem was addressed from four directions: building an understanding of software curation and preservation from various viewpoints (e.g., video games, software engineering), building a conceptual model of software preservation to guide service decisions, implementing software-related services, and documenting and evaluating the work to build expertise and establish a standard service level.

Received 19 February 2018 ~ *Accepted* 20 February 2018

Correspondence should be addressed to Fernando Rios, University of Arizona Libraries, Tucson, Arizona, United States. Email: frios@email.arizona.edu

An earlier version of this paper was presented at the 13th International Digital Curation Conference.

The *International Journal of Digital Curation* is an international journal committed to scholarly excellence and dedicated to the advancement of digital curation across a wide range of sectors. The IJDC is published by the University of Edinburgh on behalf of the Digital Curation Centre. ISSN: 1746-8256. URL: http://www.ijdc.net/

 doi:10.2218/ijdc.v13i1.608

Introduction

The work described in this paper was carried out while the author was with the Sheridan Libraries at Johns Hopkins University.

Research data management (RDM) services provided by libraries at many large research universities are well established (Ray, 2014; Soehner, Steeves and Ward, 2010). Although the specifics vary across institutions (Fearon, Gunia, Lake, Pralle and Sallans, 2013), those that offer these services provide some level of support for their researchers around data management planning, support for data management best practices (e.g., organization, workflows, storage), and support for data sharing and publication (e.g., identifying a suitable data repository, identifying publisher data sharing requirements, data de-identification). However, these services can be so data-oriented that they do not give much consideration to the software created as part of research activities (research software). This contributes to the treatment of software as a second-class research output.

Being able to examine the materials and methods used in research is critical for others to be able to recreate the analysis and results that support research claims. This act of reproducing results is a cornerstone of the scientific process. As a result, in the context of e-research, having a record of the data used or produced along with the tools used to work with the data is paramount. Although data management professionals have increasingly become aware that data and the associated software written to collect, process, analyze, and visualize that data are equally important in the context of capturing a complete picture of scholarship (Matthews, Shaon, Bicarregui and Jones, 2010; Chue Hong, 2012; Hettrick, 2016), RDM services often seem to treat software (both executables and source code) and data in the same way. This is evidenced by the recent survey (Hudson-Vitale et al., 2017) of Association of Research Libraries (ARL) member institutions about current staffing and infrastructure for data curation which showed that software is still commonly treated as second-class to data[1]. Efforts from the Software Preservation Network (Meyerson et al., 2017; Rios, Almas, Chassanoff, Contaxis, Jabloner, 2017), the Curating for Reproducibility consortium[2], and others[3] have begun to address the problem of bringing best-practices of software curation and preservation to RDM. Efforts such as those from the PERICLES[4] project and the Software Sustainability Institute[5] are also addressing the problem of software in RDM at the national/international, institutional, and individual levels. However, as evidenced by the software preservation and sustainability birds-of-a-feather session at the 12th International Digital Curation Conference, the outcomes of those efforts may take some time to become more widely adopted (Cope, 2017).

This paper presents how the data management service unit at Johns Hopkins University (JHU) addressed the lack of emphasis on software and source code in existing RDM services. The primary goal was to address the lack of emphasis on software and source code in existing offerings. The general approach was to (a)

1 In defense of the survey responders, code reviews, support for emulation, and implementing software registries were highly ranked in terms of aspirational goals for future service.
2 Curating for Reproducibility: http://cure.web.unc.edu
3 E.g., the Earth Science Information Partners (ESIP) Software Assessment Guidelines: https://esipfed.github.io/Software-Assessment-Guidelines/
4 PERICLES: http://pericles-project.eu
5 Software Sustainability Institute: https://www.software.ac.uk

determine the current practice with respect to research software in RDM, software preservation, and reproducibility, (b) use this information to identify knowledge and service gaps, (c) fill some of these gaps by incorporating software curation and preservation within existing data-oriented services, given certain operational constraints, and (d) document and evaluate the workflows and establish a standard service level.

Software as a Research Output

Before continuing, it is illuminating to consider why software should be treated differently from data and why it is important to do so. As noted by Katz et al. (2016), the main difference between data and software in the context of research is that of action: "software generally performs a function upon something (e.g., software processes data), while data generally has a function performed upon it (e.g., data is processed by software)". In other words, "we are more interested in what software does rather than what software is" (Matthews, Shaon, Bicarregui and Jones, 2010).

When dealing with software in a RDM context, data and software are often lumped together which is convenient but problematic on several fronts. First, if software is not considered separately from data, it may hinder the ease with which others (including the creators themselves) can find, understand, reproduce, and reuse not only the software, but the entire body of research. This limits the overall impact potential of the work. As an example, consider a situation where software is not given separate consideration in drafting a data management plan. The lack of attention in the resulting plan may carry through to bad practices in documenting and organizing code, which will then make it more difficult at the time of publishing to have data and code packages which adhere to FAIR (findable, accessible, interoperable, reusable) principles (Wilkinson et al., 2016). Another example relates to metadata. Metadata for datasets has been given much consideration (e.g., DataCite) and it is often used to describe software. However, software-specific metadata provides richer descriptions and efforts to bring it to the forefront are ongoing.[6]

Second, lumping data and code together discourages explicitly thinking about crediting those individuals involved in creating the software. These individuals include not only developers but also testers, bug reporters, documenters, and others that contributed to the software in some way. This lack of credit may decrease the incentive for future contributions.

Third, in the context of data sharing policies and open access discussions, if software is not given first-class consideration by funders, publishers, and institutions, it sets a bad example for those receiving funding from, publishing in, or working for said entities to also treat software on the same level as data and publications. In other words, consideration of software as integral to research (and therefore worthy of first-class, FAIR treatment), while acknowledging its commonalities with and differences from data, should take place not only from the bottom but also from the top.

6 See the introduction of software elements in DataCite (DataCite Metadata Working Group, 2017; Starr, 2017), the work of the CodeMeta project (Jones et al., 2017), and the Software Preservation Network (Meyerson et al., 2017).

Motivation, Objectives and Initial Challenges

Data management services provided by the Johns Hopkins University libraries[7] consist of providing consultative support to researchers around data management planning, data sharing, best practices for working with data (storage, encryption, sensitive data, organization), providing a data archiving service, and training/education for data management best-practices. Prior to the work presented here, the services provided were entirely focused on research data, though there was recognition that software also had a role to play. Despite this, software had not been explicitly considered in any of the service offerings. The emerging thrust in the data management and open research communities regarding transparency and reproducibility and the important role that research software plays initiated conversations into how research software could be incorporated into the data management service unit's existing offerings. Specifically, the notion of software archiving (analogous to data archiving) was a catalyst.

The objective of the project was to either extend existing data services to explicitly consider software or develop new ones. At a high level, the achieving this was straightforward: First, identify what was currently being done in the area of software curation, preservation, and archiving; investigate best practices in research transparency and reproducibility as they relate to software; distil the findings to ones which are relevant in an RDM context; determine what needed to be done to implement those which are both feasible and impactful; and increase the unit's expertise with respect to things like software curation, preservation, and the reproducibility of computational results for the purpose of supporting software-related RDM services.

The first challenge was determining where to even begin the process. The initial lack of expertise within the unit with respect to the landscape of research software as it relates to RDM and the challenges involved in developing and using software in a research environment, combined with the nascent state of research reproducibility best practices and tools meant that identifying an initial point of entry to expanding services would take some time. Addressing this challenge would involve a broad examination of material located at the intersection of data management, software preservation and curation (including theory, tools, and practice), credit for research code, metadata standards for software, the research reproducibility landscape, and tracking the rapidly evolving communities in these spaces.

The second class of challenges were resource constraints. These consisted of limited resources for raising the unit's level of expertise on such a wide variety of topics, building a prototype workflow, developing software-related services that could be plugged into existing data services with minimal effort and cost, and implementing, piloting, and evaluating them.

Developing Services

Addressing the objective meant that each of the unit's primary service areas (consulting, archiving, and training/education) needed to be examined for places where the workflow could/should be extended to include research software. This section describes what, why, and how services incorporating research software curation were implemented.

7 Data Management Services – Johns Hopkins University: http://dms.data.jhu.edu

Knowledge Building and Planning

As the first step, a comprehensive review of the literature relevant to research software preservation, reproducibility, and sharing was undertaken (Rios, 2016a). The review covered not only the scientific viewpoint but also software preservation and curation in the cultural heritage domain (e.g., video games, historical software). Topics included motivations and challenges in software preservation and curation, approaches and tools for preservation, software engineering, an overview of software metadata, open source code sharing practices, publishing, and intellectual property considerations. This review was meant to provide an overview of the landscape so as to be able to identify determine current practice, identify service gaps, and find tools/approaches with which to potentially address them. If such tool/approaches did not exist, then the review would inform what would need to be done to fill service gaps, at least in principle.

In order to translate the information from the review into a form that could be more easily used for planning, the stages and processes of developing research software were mapped onto a high-level view of approaches to software preservation and RDM. This resulted in the Pathways of Research Software Preservation (Rios, 2016b), shown in Figure 1. The figure was used over a number of group planning sessions to determine what service gaps existed and what services could be feasibly offered. The details of how this was done are presented in Rios (2016b). In summary, different preservation scenarios from the Software Preservation Benefits Framework (Chue Hong, Crouch, Hettrick, Parkinson and Shreeve, 2010) were randomly assigned to session participants. Subsequently, each person was asked to use Figure 1 to connect preservation activities across the software phases that best addressed each assigned scenario. Each person then presented to the group their justification for choosing a particular path and a group discussion followed. One of the first things that came out of this exercise was a decision to limit ‘research software’ to include only software that was created by researchers (e.g., scripts, simulations, analysis tools, etc.). Although software not created by researchers themselves (e.g., third-party libraries, commercial programming environments, etc.) were not totally excluded, not focusing on such software avoided potential rabbit holes such as a commitment to capture many levels of dependencies, and potential legal hurdles involving copyrighted software.

 doi:10.2218/ijdc.v13i1.608

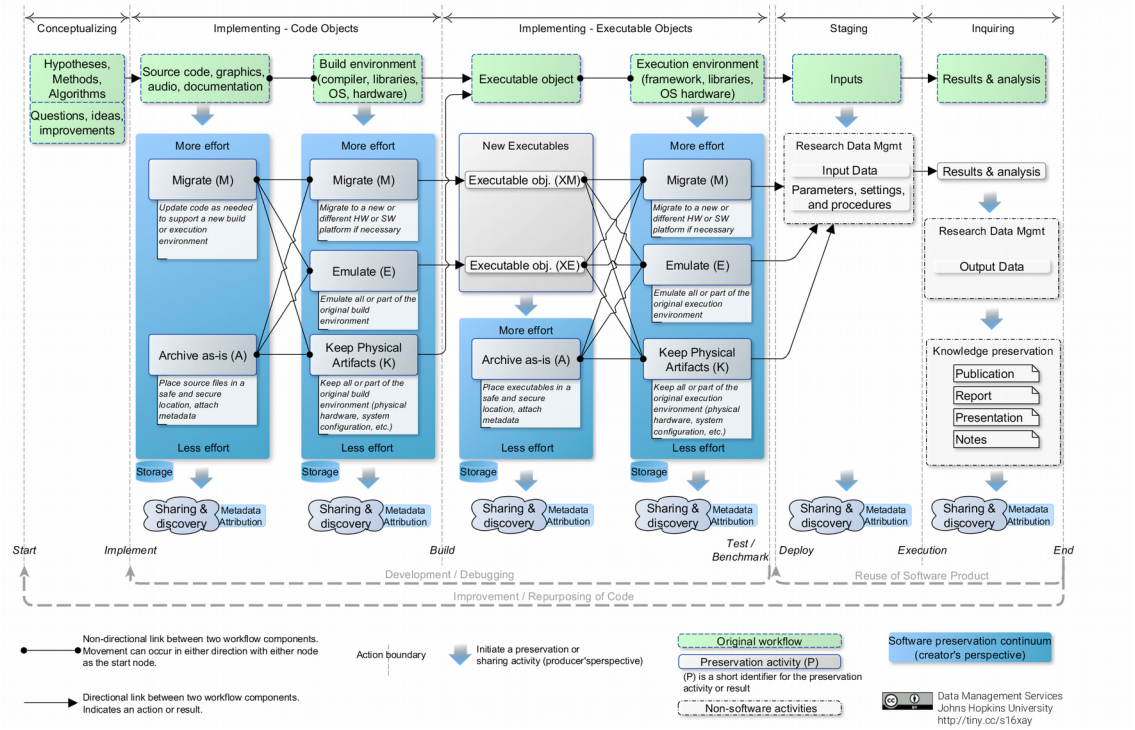

Figure 1. The Pathways of Research Software Preservation, reproduced from Rios (2016b).

The final result was a view of service gaps related to research software preservation and archiving, and a target set of services which could be expanded within the project constraints. The identified services were: archiving of research software in the JHU Data Archive[8] (and associated workflows for doing so), including software in data management planning in a way that minimized the additional work/information required from researchers, expanding the team's consulting expertise to include research software, and expanding training material to include software related aspects of data archiving and sharing.

Archiving

Developing workflows for archiving research software was one of the primary ignitors for expanding data services to include software. The exercise using Figure 1 served to solidify the primary use case: supporting funder and publisher requirements for sharing all research materials. Although some funders and many publishers do not yet require sharing of code, many others do as in conjunction with data sharing. Therefore, accommodating deposit of code so it not only meets this need but also doing it in a way that enhances its impact was deemed important.

Conceptually, the software archiving workflow is quite simple as it was developed to mirror the existing data archiving workflow, which at JHU takes the form of mediated deposit into the JHU Data Archive (Dataverse-based[9]). In summary, the workflow involves researchers providing basic information about the software, obtaining the data and code, and ingesting into the Data Archive.

The primary challenges in implementing the software-aware workflow were: structuring dataset deposits given that it is possible for researchers to provide a single

8 Johns Hopkins Data Archive: https://archive.data.jhu.edu/
9 The Data Archive is a self-hosted instance of Dataverse 4: https://dataverse.org/

'dataset' containing different combinations of data and software, and determining exactly how software should be described.

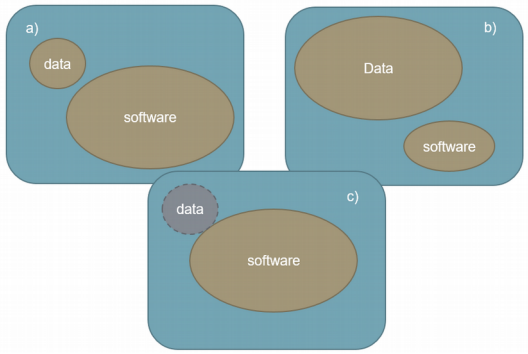

Figure 2. A conceptualization of the three different categories of software deposits considered in the archiving workflow. a) Data + software (software-focused), b) data + software (data focused), c) software-only (with possible test data included).

In regards to the first challenge, Figure 2 shows three different kinds of software-containing datasets that are possible: datasets that contain data and software where the *software* is the main research output and data plays a supporting, but still important, role (e.g., a novel algorithm with novel study data, as in Figure 2a), where *data* is the main output with software playing a supporting role (e.g., a dataset with associated non-novel plotting, analysis/cleaning scripts, as in Figure 2b), and datasets where the software is the primary focus (e.g., a novel general purpose tool meant for wide release that may have some sample data, as in Figure 2c). A fourth case where both software and data are novel and of equal importance can be handled by combining Figure 2a and b. The cases are differentiated because they have different needs when it comes to archiving and sharing. For instance, the case of Figure 2c might need the ability to track and cite different versions of the software and may be associated with multiple publications. On the other hand, the cases of Figure 2a and 2b might be self-contained studies in which the code may not ever be updated. An additional complication is that each 'software' bubble in Figure 2 could actually consist of multiple, interrelated subcomponents. Experiments using the hierarchical nature of the Dataverse data model – datasets exist within containers called 'Dataverses' which can be nested in other Dataverses – revealed that although using the nested structure to represent the various cases and subcomponents made sense conceptually, in practice the extra structure just cluttered the Dataverse interface and made the datasets harder to download and understand. In addition, since each Dataverse dataset receives its own identifier, splitting up data, code, and associated versions and subcomponents into separate datasets meant that multiple identifiers would correspond to pieces of the same body of research. It was decided that this would be highly confusing and having everything under a single identifier would be better (at the expense of precision of identification). As a result, all three cases are handled using a single Dataverse dataset and the differentiation between cases can be inferred by the information entered in a custom software metadata block in Dataverse.

To provide guidance for addressing these complexities, the following rules of thumb were adopted by the group when eliciting information from researchers so that the complexities were minimized:

- Describe software at the most general/coarsest level possible which still allows for verifying the claims made in the research,
- Include enough information to allow others to actually reuse the software,
- Include enough information so that others can give you credit.

Concept	Description	Concept	Description
SoftwareTitle *Required (1)*	Name of the software	**Version** *Optional (1)*	Software version. Some software (e.g., one-off programs) may not require a version number
SoftwareIdentifier *Required (1)*	E.g., DOI or other PID	**ObjectType** *Optional (1)*	The type of software artifact being described (source code, executable, both)
License *Required (1)*	Licensing terms.	**ProgrammingLanguage** *Optional (1+)*	The programming language used.
SoftwareContributor *Required (1+)*	Individuals or organizations that contributed (e.g., programmers, testers, documenters, etc.)	**CodeRepositoryLink** *Optional (1)*	Link to a version controlled software repository where the software is developed (e.g., GitHub, SourceForge etc.).
ContributorId *Optional (1+)*	A PID associated with each SoftwareContributor (e.g., ORCID etc.)	**RelatedObjects** *Optional (1+)*	Related papers, website, documentation, other software (except dependencies).
Description *Required (1)*	Human-readable synopsis	**Dependencies** *Optional (1+)*	Hardware (e.g., instruments, CPU, etc.), operating system, environment (e.g. Python 2.7, etc.) libraries/other SW.
datePublished *Required (1)*	Date this software was published in the archive	**InteractionMethod | Function | I/O** *Optional (1+)*	GUI, command line, etc. | General functionality (plotting, analysis, data collection, simulation, etc.) | Inputs and outputs.

Figure 3. Metadata for software, reproduced from Rios (2017). The numbers in parentheses indicate the number of entries allowed, e.g. there can only be one SoftwareTitle but there can be one or more SoftwareContributors. Several of the fields such as ProgrammingLanguage and License are controlled vocabularies.

Figure 3 shows the custom metadata block that was added to Dataverse to support richer software description. The required fields were determined by surveying existing software metadata (see Rios, 2016a) and finding the ones that appeared most often across the surveyed schemas. The optional fields were also obtained from the survey but were selected according to perceived utility (based on conversations with researchers and team expertise), ease of obtaining the required information, and ability to play a role in future linked data efforts.

The process for actually obtaining the required metadata from researchers is as follows. The required fields except datePublished (entered by the consultant as the date the deposit was made) and softwareIdentifier (an automatically assigned DOI) are obtained by extending an existing 'data deposit form' normally given to researchers to fill out prior to depositing their data. This form collects required dataset information such as title, authors, funding sources and contact information, along with some optional information, such as related publications, and file-level descriptions. Software-related additions are purposefully minimal: software description, source code repository (if available), software contributors (if different from the author list), and license. The form encourages the use of the MIT license but researchers can choose any other license as needed. After receiving the deposit from the researcher, the information is transferred into the custom metadata block in Dataverse. To fill in the optional software-related fields in the block, the consultant harvests the information in the free-text description fields from the deposit form, the information present in any source code repositories, and any other readme files contained in the dataset provided by the researcher. In the piloting phase, it was found that although some fields are easy for the consultant to complete, such as the programming language(s) and whether an object is source code or compiled executable, for others, such as dependency information, it was often only

possible to indicate high-level dependencies (e.g., MATLAB, Python 2.7) unless the information was explicitly provided by the researcher.

Consulting

Consulting services, by their nature, rely on the consultant's expert knowledge about a particular topic. To successfully expand consulting services to include research software, consultants should be knowledgeable on things like: which data repositories accept software and what are their capabilities for doing so? What best practices should be followed to ensure software is more easily preserved and the associated body of research more reproducible? What licenses are available for releasing research software? How are intellectual property issues the same or different than those for data? How, why, and where can one publish research software?

In order to build this expertise within the group, a series of bi-weekly meetings were held over the course of the project. Although some of these meetings were used for planning and iteration (e.g., for the previously mentioned planning exercise using Figure 1), many were organized in the form of an interactive hour-long seminar in which a relevant topic in software preservation, research reproducibility, or open science was presented and discussed.

Although these meetings served to introduce a topic and attendees found them interesting, the amount of information that needed to be assimilated was overwhelming at times. To mitigate this, the information was captured in internal reports that contained all the relevant information in a way that was tailored to fit with how consultants work with researchers on data consultations. However, these reports were still difficult to digest. To help with this, a quick-reference resource was designed, containing the information deemed most important for consulting. The end result took the form of a graphical 'I Need To…' reference chart (Rios, 2017) with pointers to key elements which could be further researched by the consultants as needed (e.g., via the reports).

Based on feedback from the consultants, the best approach for learning the material was by providing feedback and iterating on the creation of educational material targeted to researchers themselves.

Education and Training

Although it was one of the final aspects to be addressed, the development of educational resources targeted at researchers was one of the most visible outputs of expanding services to include research software. The reason for this is that the material created embodied all of the experience gained by expanding the other service areas and was presented in forms that were widely accessible, such as in-person and online training sessions, and via web resources. The general approach to presenting material on the role of software in enhancing reproducibility and openness was to frame it in the context of increased research efficiency, more citations, more exposure, and higher impact. The outputs were a set of online training modules, a checklist for software archiving, and links to external web resources.

Guidance for enhancing research reproducibility via good software practices was presented in the form of a series of publicly accessible online modules totalling 22 minutes in duration[10]. Apart from introductory material explaining the what and why around research software, topics were grouped into six modules (documentation,

10 Planning for software reproducibility and reuse online course: http://dms.data.jhu.edu/training/online-training/software-online-training/

organization and automation, version control and quality assurance, context and credit, licensing, and archiving) and included: best practices for writing, documenting, and organizing code, understanding why version control is good, linking code to research results, making code publishable and citable, intellectual property issues, and the why and how of preserving software over time.

In order to give more clarity to the process of creating a high-quality software package for archiving, a checklist[11] with the most important considerations for doing so was created. These considerations include thinking about the target audience and thinking about the different pieces of information needed for the metadata collected during the deposit process.

Lessons Learned

The project met the objectives of developing software archiving services that fit alongside existing data services. In terms of software archiving in the data repository, the prototype workflows and metadata structures were successful as evidenced by enhanced visibility of software in datasets in the JHU Data Archive. However, some tweaking of the workflow was required as a result of piloting. Originally, a workflow which incorporated the OntoSoft (Gil, Ratnakar and Garijo, 2015) software discovery portal was trialled. OntoSoft was seen as ideal for capturing software information because the OntoSoft ontology captures many aspects of research software in a structure which embodies linked data principles and exposes them via a convenient user interface. However, difficulties in linking the software/dataset located in Dataverse with the associated entry in OntoSoft in a satisfactory way resulted in abandoning the idea of using OntoSoft as a metadata store.

Quantifying the success of knowledge-building is more difficult but informal conversations with team members revealed that they felt that at the end of the project, they had a stronger capability of supporting software-related questions in the context of RDM. Part of this confidence came from the great deal of effort spent documenting workflows and creating and refining educational material. The latter was mentioned by the team as one of the most helpful ways to learn about the subject matter. In terms of broader impact, the online modules in particular, were the most successful as a result of the visibility they received on social media at the time of release. This shows that there is a demand for easy to digest, practical advice on working reproducibly with software.

As is often the case, the most challenging part of the process was not technical but human in nature. For example, generating ideas for how the archiving workflow might look like was relatively easy and ‘fun’ (enabled by planning tools such as Figure 1). However, actually developing and refining the workflows took much longer due to the need to dive deep into the details and consultants’ limited time each month to work on the project. To make the process move more smoothly, developing rough prototypes sooner would have been beneficial so as to be able to iterate over a longer period of time.

In the course of developing the services, the similarities and differences of software with data became acutely visible, especially after the planning exercises. Reflecting on the outcomes of integrating software into RDM services, it is obvious that there exists a vast chasm between the effort and resources needed to support research software in a

11 Software archiving checklist: http://dms.data.jhu.edu/data-management-resources/publish-and-share/software-archiving/software-archiving-checklist/

dataset-like way and the capability of truly capturing the essence of research software. For example, after incorporating software into the existing archiving workflow, it became readily apparent that the effort required for providing services for archiving software source code (including metadata, structuring of the code and associated datasets into an archival package, and providing educational support for doing so) is orders of magnitude smaller than the effort needed to provide services (e.g., technical platforms, consulting) for actually supporting software re-execution now and in the future. Although there are solutions for doing so, enabling their success requires effective governance, policy, and, critically, viable business models. Cyberinfrastructure projects falling under the umbrella of 'science gateways'[12], examples of which include XSEDE (Towns et al., 2014) and CyVerse (Goff et al., 2011; Merchant et al., 2016), are tackling the issue but they are specifically resourced to do so at funding levels much higher than available to library-based RDM services. As a result, providing truly software-oriented services will likely remain out of reach for library-based RDM consulting units unless partnerships are forged with those that have the capacity to cater to the aspects of research software that truly separate it from data.

Acknowledgements

The author would like to acknowledge Barbara Pralle, Jonathan Petters, Reid Boehm, Chen Chiu, Dave Fearon, Tim DiLauro, Sayeed Choudhury, and Eliott Metsger for their contributions during the course of this work. Support was provided in part by a Council for Library Information Resources (CLIR) postdoctoral fellowship.

References

Chue Hong, N. (2012). Digital preservation and curation: The danger of overlooking software. In *The Preservation of Complex Objects: Volume 1 – Visualisations and Simulations*. Retrieved from http://www.research.ed.ac.uk/portal/en/publications/digital-preservation-and-curation-the-danger-of-overlooking-software(200ad653-3e21-49d2-beae-4dcb581ea934).html

Chue Hong, N., Crouch, S., Hettrick, S., Parkinson, T., & Shreeve, M. (2010). *Software preservation benefits framework*. Software Sustainability Institute.

Cope, J. (2017). Software preservation and sustainability. Retrieved from http://www.dcc.ac.uk/news/software-preservation-and-sustainability

DataCite Metadata Working Group. (2017). *DataCite metadata schema for the publication and citation of research data v4.1*. DataCite. doi:10.5438/0015

Fearon, D.J., Gunia, B., Lake, S., Pralle, B.E., & Sallans, A.L. (2013). Research data management services, SPEC Kit 334. Retrieved from http://publications.arl.org/Research-Data-Management-Services-SPEC-Kit-334/

12 Internation Coalition on Science Gateways: http://www.icsciencegateways.org/

Gil, Y., Ratnakar, V., & Garijo, D. (2015). OntoSoft: Capturing scientific software metadata. *Proceedings of the Eighth ACM International Conference on Knowledge Capture*, Palisades, NY, ACM Press. doi:10.1145/2815833

Goff, S.A., Vaughn, M., McKay, S., Lyons, E., Stapleton, A.E., Gessler, D., … Stanzione, D. (2011). The iPlant Collaborative: Cyberinfrastructure for plant biology. *Frontiers in Plant Science, 2*. doi:10.3389/fpls.2011.00034

Hettrick, S. (2016). *Research Software Sustainability: Report on a Knowledge Exchange Workshop*. Retrieved from http://www.knowledge-exchange.info/event/software-sustainability

Hudson-Vitale, C., Imker, H., Johnston, L.R., Carlson, J., Kozlowski, W., Olendorf, R., & Stewart, C. (2017). Data curation, SPEC Kit 354. Retrieved from http://publications.arl.org/Data-Curation-SPEC-Kit-354/

Jones, M.B., Boettiger, C., Mayes, A.C., Smith, A., Slaughter, P., Niemeyer, K., … Goble, C. (2017). CodeMeta: An exchange schema for software metadata. KNB Data Repository. doi:10.5063/schema/codemeta-2.0

Katz, D.S., Niemeyer, K.E., Smith, A.M., Anderson, W.L., Boettiger, C., Hinsen, K., … Rios, F. (2016). *Software vs. data in the context of citation* (No. e2630v1). PeerJ Inc. doi:10.7287/peerj.preprints.2630v1

Matthews, B., Shaon, A., Bicarregui, J., & Jones, C. (2010). A framework for software preservation. *International Journal of Digital Curation*, *5*(1), 91–105. doi:10.2218/ijdc.v5i1.145

Meyerson, J., Vowell, Z., Hagenmaier, W., Leventhal, A., Roke, E.R., Rios, F., & Walsh, T. (2017). The Software Preservation Network (SPN): A community effort to ensure long term access to digital cultural heritage. *D-Lib Magazine*, *23*(5/6). doi:10.1045/may2017-meyerson

Merchant, N., Lyons, E., Goff, S., Vaughn, M., Ware, D., Micklos, D., & Antin, P. (2016). The iPlant Collaborative: Cyberinfrastructure for enabling data to discovery for the life sciences. *PLOS Biology, 14*(1), e1002342. doi:10.1371/journal.pbio.1002342

Ray, J.M. (Ed.). (2014). *Research data management: Practical strategies for information professionals*. West Lafayette, Indiana: Purdue University Press.

Rios, F. (2016a). Preserving and sharing software for transparent and reproducible research: A review. doi:10.17605/OSF.IO/D4KEF

Rios, F. (2016b). The pathways of research software preservation: An educational and planning resource for service development. *D-Lib Magazine, 22*(7/8). doi:10.1045/july2016-rios

Rios, F. (2017). A toolbox for curating and archiving research software for data management specialists. Paper presented at the 12th International Digital Curation Conference, Edinburgh, UK. Retrieved from http://www.dcc.ac.uk/webfm_send/2398

Rios F., Almas B., Chassanoff A., Contaxis N. & Jabloner P. (2017). Exploring curation-ready software: Improving curation-readiness. doi:10.17605/OSF.IO/T9G3Q

Soehner, C., Steeves, C., & Ward, J. (2010). *E-Science and data support services: A study of ARL member institutions*. Association of Research Libraries. Retrieved from https://eric.ed.gov/?id=ED528643

Starr, J. (2017). New DataCite metadata updates support software citation [website]. Retrieved from https://blog.datacite.org/metadata-schema-4-1/

Towns, J., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, A., … Wilkens-Diehr, N. (2014). XSEDE: Accelerating scientific discovery. *Computing in Science and Engineering, 16*(5), 62–74. doi:10.1109/mcse.2014.80

Wilkinson, M.D., Dumontier, M., Aalbersberg, Ij.J., Appleton, G., Axton, M., Baak, A., … Mons, B. (2016). The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, *3*, 160018. doi:10.1038/sdata.2016.18