Proceedings of the Twelfth IEEE Conference on eScience, Baltimore, MD, November 2016.

OntoSoft: A Distributed Semantic Registry for Scientific Software

Yolanda Gil, Daniel Garijo, Saurabh Mishra, Varun Ratnakar
Information Sciences Institute
University of Southern California
Los Angeles, CA, USA
gil@isi.edu, dgarijo@isi.edu, saurabhm@usc.edu, varunr@isi.edu

***Abstract*— OntoSoft is a distributed semantic registry for scientific software. This paper describes three major novel contributions of OntoSoft: 1) a software metadata registry designed for scientists, 2) a distributed approach to software registries that targets communities of interest, and 3) metadata crowdsourcing through access control. Software metadata is organized using the OntoSoft ontology along six dimensions that matter to scientists: identify software, understand and assess software, execute software, get support for the software, do research with the software, and update the software. OntoSoft is a distributed registry where each site is owned and maintained by a community of interest, with a distributed semantic query capability that allows users to search across all sites. The registry has metadata crowdsourcing capabilities, supported through access control so that software authors can allow others to expand on specific metadata properties.**

Keywords—software registries, software metadata, scientific software, software catalogs, software repositories

I. INTRODUCTION

The software developed by scientists embodies important scientific knowledge that should be explicitly captured, curated, managed, and disseminated. Software captures mathematical models, statistical analyses, and causal reasoning that are used to generate new results. Scientists recognize the value of sharing software to avoid replicating effort and to inspect and reproduce results from others. In addition, recurring issues of provenance and uncertainty in the context of data could be better addressed with improved treatment of software: one of the best ways to understand data is to look at the software that uses it or generates it.

A major issue for scientific software reuse is the dissemination and documentation of existing codes. Although code repositories already exist and are used by many scientists, they typically contain basic metadata such as authors, license but lack appropriate metadata to facilitate discovery and reuse.

A second major issue in scientific software sharing is the limited sharing of codes used in scientific publications. While the loss of "dark data" in science is well recognized [1], we see an analogous problem in the pervasive loss of "dark software". Many scientists do not share their software, because they are unaware of its value, or they do not know how, or they are worried about not getting proper acknowledgment, or they do not see its value. Studies show that scientists spend between 60% to 80% of a project's effort collecting and preparing data before doing new science (e.g., [2]). This would indicate a significant overhead in developing software for data preparation that is only rarely shared and rarely reused. A common concern is the relatively lower quality of such software, since it is typically not written with robustness or generality in mind and scientists do not want their reputations tarnished [3]. Scientists should be aware of the value of their software, and have the means to share it easily and worry-free.

A third major issue for scientific software sharing is adoption and trust. Scientists should be able to recognize whether they can trust software developed by others since they rely on it to do science. Scientists today have no access to the kinds of community ratings that help assess quality and reuse.

This paper reports on OntoSoft, a distributed semantic software registry aimed to improve scientific software stewardship through: 1) an ontology for software metadata, which organizes information about software in metadata categories that are designed with scientists in mind; 2) metadata crowdsourcing capabilities, where software authors can control permissions for others to edit specific metadata entries; and 3) a distributed architecture where each site can be overseen by a community of practice but metadata is exported and can be queried across all sites.

The paper begins discussing relevant work on characterizing software and how software is described in code repositories today. It introduces the requirements for OntoSoft based on feedback from scientists. The main section of the paper describes the architecture of the OntoSoft registry and its implementation, followed by conclusions and future work.

II. SOFTWARE REPOSITORIES AND OTHER RELATED WORK

In this section we discuss releated work on capturing, describing, and sharing scientific software. In addition, we discuss the kinds of metadata used in code repositories, both general-purpose and specific to scientific domains. All this work has informed the design of OntoSoft.

A. Capturing Information about Software

The Core Software Ontology (CSO) and the Core Ontology of Software Components (COSC)[1] [4] extend the DOLCE ontology [5] to describe software components and web services. These ontologies were designed to describe large software systems, so their requirements include the

This work was supported by the U.S. National Science Foundation under award ICER-1440323.

[1] http://cos.ontoware.org

accessibility of the software components, middleware services, execution failures, and composition of software. CSO formalizes concepts related to software and data, and includes both software components and services. COSC extends CSO to define software components further, and includes notions such as interaction protocols and taxonomies. There are several major departures from our goals in OntoSoft. First, these ontologies are very formalized and axiomatized based on the *Descriptions* and *Situations* ontology design pattern that captures how states change when actions are executed. This is not an aspect of software that is central to a software catalog. Second, such formalizations would place restrictions on what users must specify, which would require users to understand logic inference in ways that an average scientist would not know. Finally, they focus on complex software systems, rather than in end users who are scientists and need to describe software for other scientists.

Chue-Hong [6] proposes a framework for capturing information about software that promotes reusability. The framework consists of four levels: absolute minimum (L1), useful minimum (L2), pragmatic minimum (L3), and good minimum (L4). Each level contains information that is split into five categories: license (legal constraints), availability (discovery and accessibility), quality (understanding functional and non-functional characteristics), support (communicating with original developers), and incentive (rewarding developers). These levels build on one another, and the framework includes two additional levels at the extremes that represent theoretical minimum (L0, insufficient for reusability) and idealistic minimum (L5, too much required from the developer). For example, in the license category L0 requires that the software has any license, L1 that the license allow reuse, L3 that the license allows modification as well as reuse, and L4 that it is an approved open source license allowing modification and reuse (L2 and L5 are omitted for this category). The framework is only described informally, and is not specified as a model or ontology. The framework is complementary to OntoSoft, in that it could be incorporated as an extension by specifying required properties and values in each of the levels.

WICUS[2] [7] is an ontology for describing execution requirements of workflows. It has a complementary focus, and could be used in combination with OntoSoft as an extension to specify runtime requirements.

The TIMBUS Ontologies[3] [8] propose a model to preserve and ensure the availability of business processes and their computational infrastructure, aligned with enterprise risk and business management. They also propose a semantic approach to describe execution environments of processes, which could be combined with OntoSoft. Even though TIMBUS has studied the applicability of their approach to scientific software, it is focused on business processes.

The Software Ontology (SWO)[4] is a heavyweight ontology that aims to help describing software used by the curation and data preservation community in domains ranging from text bioinformatics to social sciences. SWO extends some of the Open Biomedical Ontologies (OBO)[5], like the Basic Formal Ontology (BFO)[6] as foundational ontology and the Relations Ontology (RO)[7] to describe software relationships. It also extends the EDAM Ontology[8] for adapting different data formats and operations in the bioinformatics domain. However, the level of sophistication in formal logic of the SWO makes it less accessible to users who lack such expertise, as is the case with most scientists. Some of the SWO terms may be aligned with OntoSoft to allow describing further some aspects of software (i.e., software composition) that are outside the initial scope of OntoSoft.

CodeMeta[9] is a recent effort to map across software metadata across repositories. It includes a mapping to the OntoSoft ontology.

In [9], the authors argue for the use of software security maturity models and apply them to characterize the levels of security and reliability of scientific software. These kinds of models have not been applied to scientific software, so they are not covered in our OntoSoft ontology but could be an extension of it.

B. General Software Repositories

Software repositories are widely used by scientists. They have different features and utility. Although they allow users to describe their software and often use standard conventions for doing so, they do not use an ontology or model to organize the descriptions of the software.

General code repositories are widely used for scientific software. GitHub[10] is a repository that supports version control through the Git infrastructure. GitHub projects have a standard way to specify documentation (through readme files and collaborative Wiki pages) and licenses. BitBucket[11] is a private software repository that allows synchronizing through Git and Mercurial repositories. SourceForge[12] is a software repository for open source software projects, where users can provide an overview of the features of their project and point to the supported executables or installers for download.

Other code repositories are more focused on science, and attract contributions in many domains. CRAN[13] is an archive of code written in the R language. CRAN is built collaboratively, based on a network of ftp and web servers that store up-to-date versions of code and documentation. The only requirement for submitting new code is to fill a short form and provide documentation. Similarly, PyPI[14] is a software repository of Python codes [10], although scientific software tends to be in the SciPy repository [11] described below.

[2]http://purl.org/net/wicus
[3]http://timbusproject.net/portal/publications/ontologies
[4]http://theswo.sourceforge.net, https://softwareontology.wordpress.com/

[5]http://www.obofoundry.org/
[6]http://www.obofoundry.org/cgi-bin/detail.cgi?id=bfo
[7]http://www.obofoundry.org/cgi-bin/detail.cgi?id=ro
[8]http://edamontology.org/
[9] https://github.com/codemeta/
[10]http://www.github.com
[11]https://bitbucket.org/
[12]http://sourceforge.net/
[13]http://cran.r-project.org/
[14]https://pypi.python.org/pypi

FigShare[15] and Zenodo[16] are repositories for research artifacts (papers, blog posts, datasets, etc.) including software. Both allow users to describe all these artifacts with keywords, assign them DOIs, link to a code repository (e.g. GitHub), and add the corresponding license and publication date. An important feature of these repositories, is that they specify how to cite the software so authors can get credit.

Kaggle[17] is a site that holds competitions for data analytics where companies and government agencies post their data and pose challenges and data mining experts around the world can submit entries to the challenge. Winning solutions have to be documented through a template[18]. The template includes details about the machine learning approach taken, such as feature selection and extraction, training procedures, and formation of model ensembles. It also includes the description of inputs and outputs of codes, their functions, their runtime dependencies, and detailed instructions to run them.

Workflow systems use domain-independent languages to describe the software components that are used as steps in the workflows [12,13,14,15]. These languages capture information about how the codes need to be invoked, basic use documentation, and execution information. Workflow repositories, such as myExperiment [16] and CrowdLabs [17], do not use ontologies to capture structured software descriptions of the individual workflow steps.

C. *Software Repositories in Science Domains*

There are several repositories that have been developed for specific science domains that have different approaches and rationale as well as practical experiences with the perceived benefits and incentives of collecting software metadata. We designed OntoSoft to cover all the metadata that these repositories collect, and more.

The Community Surface Dynamics Modeling System (CSDMS) contains hundreds of codes for models for Earth surface processes [18]. CSDMS also collects a comprehensive set of metadata for software, including authors, programming languages, pointers to code, licenses, and test datasets. It also assigns DOIs to models. Other modeling frameworks in geosciences include the Earth System Modeling Framework (ESMF)[19] and the Computational Infrastructure for Geodynamics (CIG[20]. These frameworks support sophisticated model coupling capabilities to run several models in consonance, such as re-gridding to match the model scales and message passing across models to synchronize the processes they each model, which requires standard interfaces. These are aspects not covered by OntoSoft.

In [19] the authors describe the practical experiences with a software repository for astronomy, the Astrophysics Source Code Library (ASCL). A major community driver for this resource is that it gives authors the ability to cite software from the repository, and in addition it collects citations for each software entry. Each software entry is described with five fields: 1) code name, 2) brief description, 3) authors, 4) URL to download the code, and 5) unique identifier for the software (the ASCL ID). The entry also includes a link to a paper describing the software, and links to papers using the software. Through interactions with the astronomy community, it was found that users prefer to keep any metadata together with the code, that they would rather use ASCL more as a registry than as a repository (jumping from 40 codes for several years to 700 entries in just 3 years), that the type of license or the quality of the code are not as important as having them registered and indexed, and that any information that changes over time (e.g., versions) should not be captured because it is too hard to track. Due to lack of resources, their focus is on metadata that enables identifying the code accurately and without ambiguity. [19] discuss other code repositories in astrophysics that were not so successful due to lack of exposure in the community.

nanoHUB [20] is a science gateway that contains for nanotechnology software and educational materials. A license is required, and approximately one-eighth of the codes use open source licenses. Code providers are encouraged to provide documentation for first time users, test suites, and a citation for the software. A key added value of the repository is the measures of quality of the software, which are contributed by its more than 330,000 users annually. They are tracked through usage statistics and citations. In addition, reviews and questions/answers are associated with each code, as well as wish lists from users. Exposing these indirect measures of quality incentivizes code authors to support their code. Citation is supported, but [20] reports that over a hundred tools have been cited once and only fourteen have been cited ten or more times. HUBzero [21] is the framework underlying nanoHUB, and it has been used to develop web sites in different domains. One such site is the hub for the volcanology community [22]. They recommend to include benchmarks and keep track of popular codes, to document usage limitations and scope of the codes so they are not wrongly utilized, and facilitate integration within a workflow.

SciPy [11] is an open-source library of scientific python code. It includes packages for mathematical computations, plot generation, and publishing of interactive results through iPython [23]. SciPy is only accessible and understandable by python programmers.

Some workflow systems include a significant amount of codes in a domain that can be used as workflow steps, including LONI Pipeline for neuroimaging genomics [24], GenePattern and Galaxy for genomics [25], and Taverna for bioinformatics services [12]. However, the descriptions of these codes are typically only structured in terms of inputs and outputs, and other information is simply text documentation.

III. Requirements and Design of OntoSoft

There are important requirements not met by current software repositories that motivate our design for OntoSoft:

- *A software registry to complement code repositories*. There are already code repositories that offer important

[15]http://figshare.com/
[16]https://zenodo.org/
[17]http://www.kaggle.com/
[18]https://www.kaggle.com/wiki/WinningModelDocumentationTemplate
[19] https://www.earthsystemcog.org/projects/esmf/
[20] https://geodynamics.org/cig/

functionality in terms of collaborative software development, version control, and community support. However, these repositories are not integrated with one another, and their registry aspects (i.e., the metadata that describes the software entries) are not very structured because they are included in readme files and other informal means. As a result, it is hard for a scientist to find software with desired properties, such as finding a software package for k-means, in Java, with a license that allows commercialization, and that takes data in NetCDF format. In addition, many scientists have limited software skills and find code repositories hard to use. They are more amenable to sharing code through a data repository such as Zenodo. There is a need for a registry that would describe software entries with proper metadata, while pointing to a repository to download the software itself.

- *A software metadata vocabulary designed for scientists and scientific software.* The information and documentation that is available in code repositories is typically centered on software installation, rather than on software sharing. Execution information is sometimes extracted automatically (e.g., from virtualization environments). There is little guidance on how to describe software to facilitate the kind of understanding that scientists require in order to trust it and use it to do their science. This includes for example understanding the assumptions made, or the projects or publications that use the software. New approaches for capturing scientific software metadata are needed.
- *A social approach to scientific software documentation*. Users are rarely excited about providing metadata – not for datasets and not for software. In addition, when considering reuse, scientists often want to know what other scientists thought of the software as they tried to use it. This requires opening software documentation beyond software authors. This also liberates the software authors from not only having to deliver the software but also the metadata. It also allows others to specify the metadata as they read through documentation and perhaps also the literature, and reflecting their own experiences with respect to usability and quality of the code. At the same time, authors should have some control over what is said about their software to ensure it is accurate. There is a need to open software metadata to contributions from the community, not just the software authors, while retaining some control to ensure utility and quality.

These requirements stem from informal surveys of scientists done as part of the OntoSoft project[21], and have driven us to design the OntoSoft distributed semantic registry for scientific software metadata.

IV. The OntoSoft Distributed Semantic Registry for Scientific Software

The metadata captured by OntoSoft is organized through an ontology, described in [26]. The software metadata properties specified in the ontology are organized into six major categories based on information that a scientist would seek: 1) identify software, 2) understand and assess software, 3) execute software, 4) get support, 5) do research, and 6) update the software. Properties are marked as important or optional.

Figure 1 highlights the main features of OntoSoft described throughout this section. The user is shown indicators of metadata completeness. Some metadata is imported automatically from software catalogs. For example if the user specifies a GitHub site, OntoSoft will import metadata such as authors, contributors, and license. The user can export the software metadata in HTML, RDF and JSON, so they can include the metadata in their publications or attach it to their software.

A. Distributed Architecture

Each OntoSoft site has access to the content of other repositories, so software entries can be shown to users together with their source. Users can search for software based on semantic metadata properties, and get results for software in any of the OntoSoft sites.

Although OntoSoft is still a prototype under development, there are currently several OntoSoft sites deployed to describe software in different communities like Earth systems modeling, paleoclimatology, and geospace sciences, and environmental omics[22]. There are currently more than 600 software entries described in OntoSoft.

B. Crowdsourcing Metadata through Access Control Policies

Software authors can open the metadata editing selectively to others. This means that the original software developers are not necessarily responsible for providing all metadata, which can be provided by those who benefit from using the software. Software authors should be able to decide what metadata they are willing to crowdsource. For example, they may want to allow others to edit the metadata about uses of their software, but not metadata about the version releases. In some cases, permission to edit may only be given to selected contributors.

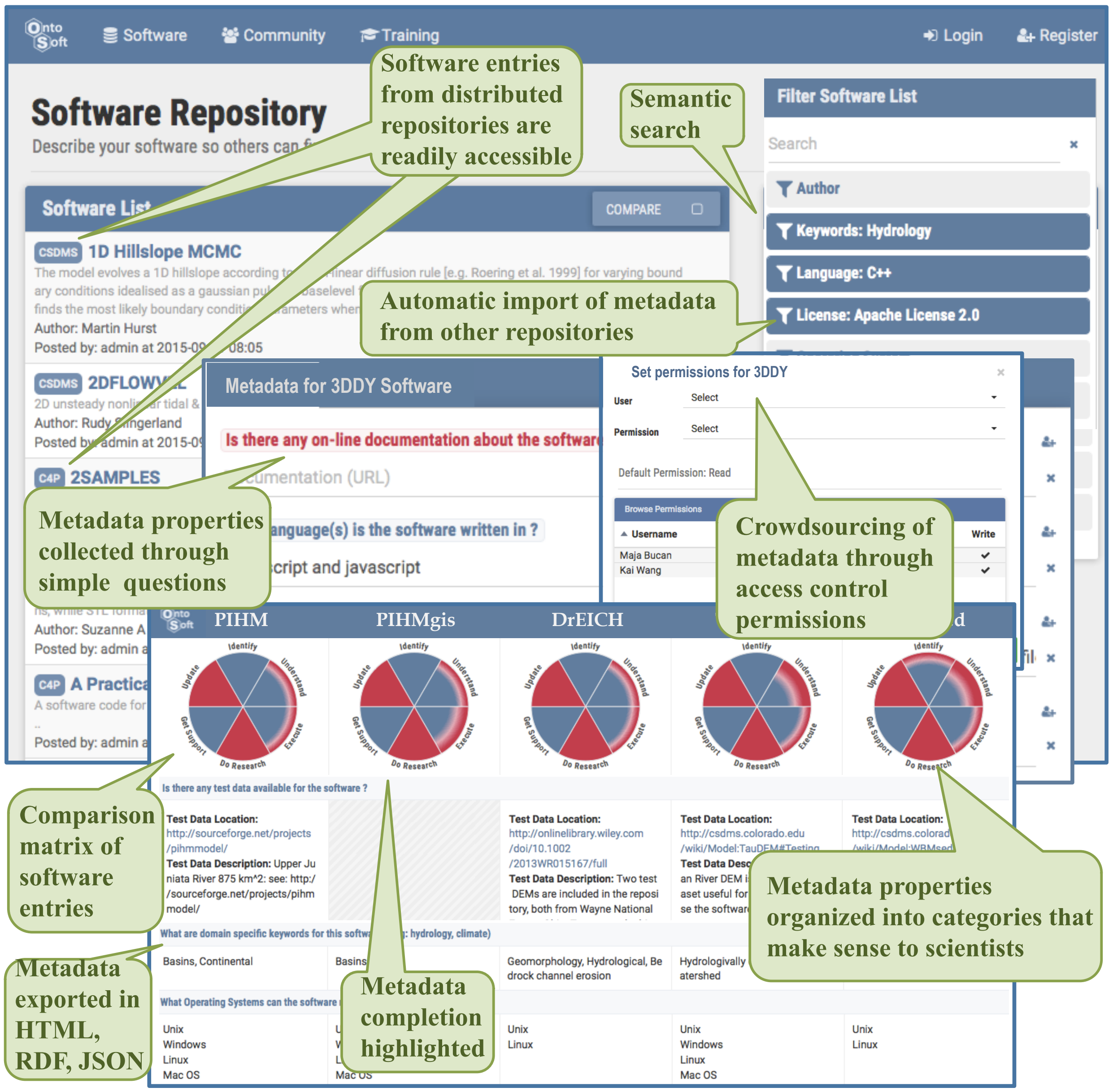
Our approach is to give authors access control mechanisms. OntoSoft extends the W3C WebAccessControl ontology.[23] Authorization is generally implemented using access control lists (ACLs), which consist of a series of access control instructions that either allow or deny permissions (such as read, write, etc.) to specified entries and their attributes. The WebAccessControl ontology incorporates these concepts by defining acl:Authorization, an abstract entity whose properties (acl:accessMode, acl:accessTo, and acl:agent) are defined in an ACL. In the ontology, acl:InformationResource can refer to either a software entry or to a metadata property of a software entry, which allows finer-grained software control.

Figure 2 illustrates this fine-grained access control. Here, a software author has created a software software1 and defined two specific permissions. Permission auth1 specifies that user user1 has read access to any metadata property of software1. A second permission auth2 specifies that user user2 has write access to the metadata property hasUseStatistics. The corresponding access control list entries are:

[21] http://ontosoft.org/

[22] http://www.ontosoft.org/portals/

[23] https://www.w3.org/wiki/WebAccessControl


Software entries from distributed repositories are readily accessible
Semantic search
Automatic import of metadata from other repositories
Metadata properties collected through simple questions
Crowdsourcing of metadata through access control permissions
Comparison matrix of software entries
Metadata properties organized into categories that make sense to scientists
Metadata exported in HTML, RDF, JSON
Metadata completion highlighted
Software
Community
Training
Login
Register
COMPARE
Identify
Understand
Execute
Do Research
Get Support
Update

Software Repository

Describe your software so others can f

Software List

CSDMS 1D Hillslope MCMC

The model evolves a 1D hillslope according to ... linear diffusion rule [e.g. Roering et al. 1999] for varying bound ary conditions idealised as a gaussian pul... baselevel ... finds the most likely boundary conditi... rameters when

Author: Martin Hurst

Posted by: admin at 2015-09 ... 08:05

CSDMS 2DFLOWV...L

2D unsteady nonli...ar tidal &

Author: Rudy ...ngerland

Posted by admin at 2015-09

C4P 2SAMPLES

Author: Suzanne A

Posted by: admin a

C4P A Practica

A software code for

Posted by: admin a

Filter Software List

Search

Author

Keywords: Hydrology

Language: C++

License: Apache License 2.0

Metadata for 3DDY Software

Is there any on-line documentation about the software

...cumentation (URL)

...anguage(s) is the software written in ?

...cript and javascript

Set permissions for 3DDY

User: Select

Permission: Select

Default Permission: Read

Browse Permissions

Username	Write
Maja Bucan	✓
Kai Wang	✓

	PIHM	PIHMgis	DrEICH		...d
Is there any test data available for the software ?					
	Test Data Location: http://sourceforge.net/projects/pihmmodel/ Test Data Description: Upper Juniata River 875 km^2: see: http://sourceforge.net/projects/pihmmodel/		Test Data Location: http://onlinelibrary.wiley.com/doi/10.1002/2013WR015167/full Test Data Description: Two test DEMs are included in the repository, both from Wayne National	Test Data Location: http://csdms.colorado.edu/wiki/Model:TauDEM#Testing Test Data Desc... an River DEM i... aset useful for ... se the softwar...	Test Data Location: http://csdms.colorad.../wiki/Model:WBMsed
What are domain specific keywords for this softw... (e.g.: hydrology, climate)					
	Basins, Continental	Basins	Geomorphology, Hydrological, Bedrock channel erosion	Hydrologically ... atershed	
What Operating Systems can the software ...					
	Unix Windows Linux Mac OS	U... W... L... Mac OS	Unix Linux	Unix Windows Linux Mac OS	Unix Linux

Fig. 1. An overview of the user interface of the OntoSoft editor for software metadata properties, highlighting its main features as a distributed architecture, its scientist-centered design, and its metadata crowdsourcing approach.

```
[acl:accessTo <http://www.ontosoft.org/portal/software/software1>;
     acl:mode acl:Read;
     acl:agent <http://www.ontosoft.org/portal/users/user1>]
[acl:accessTo <http://www.ontosoft.org/portal/software#hasUseStatistics>;
     acl:mode acl:Write;
     acl:agent <http://www.ontosoft.org/portal/users/user2>]
```

OntoSoft uses Discretionary Access Control (DAC) [27], which allows the owner of a resource to determine which users can access it. Users are given one of three roles for each software entry, which are used to set permissions: software owners, software editors, and metadata editors. A *software owner* has complete access to all the aspects of a software entry, and can make others editors of the entire entry or editors of specific metadata properties. Software owners have permissions to delete the entry, add or remove software editors, add or remove property editors, and edit any properties. A *software editor* has permission to edit all the metadata properties of a software entry. A *metadata editor* has permission to update properties they have been granted write access to for a given software entry. Figure 1 shows in a pop-up window how permissions can be granted to users. Each OntoSoft site can be configured with defaults for new entries.

C. *Querying Distributed Software Registries*

Each OntoSoft site has a public API that can be used to retrieve a list of all software entries and all the metadata for a particular software entry. The results can be in different serializations (e.g., JSON, RDF/XML). Each OntoSoft site

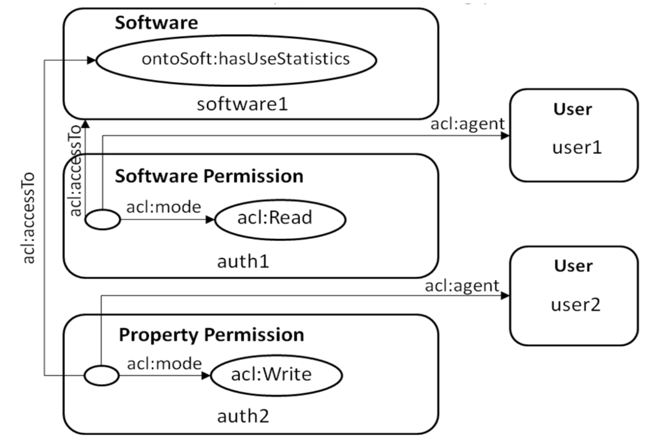


Fig. 2. Decentralized access control system for allowing different users various forms of access to resources.

shows its own software entries by default. Each site can be configured to index the software entries from other sites, aggregating all their data so it can be efficiently queried. This allows users to view, filter, and compare entries from other sites. When a user is logged into a site and selects a software entry from another site, the user is redirected to the page of that software entry in the site in which the entry was created. With this approach, the information is decentralized and each community is responsible for their own software entries while enabling others to search and compare contents from their site.

V. CONCLUSIONS

OntoSoft is a semantic registry for capturing scientific software metadata that provides the means to crowdsource software metadata while granting software owners control over who can modify their entries. OntoSoft has a distributed architecture, so that different communities can run their own sites while keeping them all interconnected. This enables a distributed query capability so that users can search software entries across different OntoSoft sites. We plan to enable cross-site user registration and access control so that users can login with the same credentials and access rights across sites. Future work also includes improving integration with existing code repositories, and developing a recommender system.

ACKNOWLEDGMENTS

We would like to thank other members of the OntoSoft project, including Scott Peckham, Chris Mattmann, Kaijian Xu, Erin Robinson, and Chris Duffy. We would also like to thank the many early adopters of OntoSoft for their feedback and comments on this work, and in particular Cedric David, Leslie Hsu, Anna Kelbert, and Sandra Villamizar.

REFERENCES

[1] Heidorn, P.B. "Shedding Light on the Dark Data in the Long Tail of Science." Library Trends, Vol. 57, No. 2, Fall 2008.

[2] Garijo, D.; Alper, P.; Belhajjame, K.; Corcho, O.; Gil, Y.; and Goble, C Common Motifs in Scientific Workflows: An Empirical Analysis.. Future Generation Computer Systems, . 2013.

[3] Barnes, N. Publish your computer code: It is good enough. Nature 467, 753, 2010. doi:10.1038/467753a

[4] Oberle D., Lamparter S., Grimm S., Vrandecic D., Staab S., Gangemi A. "Towards Ontologies for Formalizing Modularization and Communication in Large Software Systems." Journal of Applied Ontology, Vol. 1, No. 2, 2006.

[5] Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L. "Sweetening Ontologies with DOLCE." Proceedings of the 13th Conference on Knowledge Engineering and Knowledge Management (EKAW), 2002.

[6] Chue-Hong, N. "Minimal information for reusable scientific software." Presented at the Second Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE2), 2014. figshare. http://dx.doi.org/10.6084/m9.figshare.1112528.

[7] Santana-Perez, I., Pérez-Hernández, M. "Towards Reproducibility in Scientific Workflows: An Infrastructure-Based Approach" Scientific Programming, vol. 2015, 2015.

[8] Mayer, R. Miksa, T. and A. Rauber. Ontologies for describing the context of scientific experiment processes, in: 10th International Conference on e- Science, 2014.

[9] Heiland, R., Thomas, B., Welch, C. and C. Jackson. "Towards a Research Software Security Maturity Model." Presented at the First Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE1), 2013, http://arxiv.org/abs/1309.1677

[10] Terrel, A. "Sustaining the Python Scientific Software Community." Presented at the First Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE1), 2013. figshare. http://dx.doi.org/10.6084/m9.figshare.791565

[11] Jones E., Oliphant E., Peterson P., et al. SciPy: Open Source Scientific Tools for Python, 2001, http://www.scipy.org/ [Accessed 2015-03-13].

[12] Missier, P., Soiland-Reyes, S., Owen, S., Tan, W. et al. Taverna, reloaded. In 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany, 2010.

[13] Ludaescher, B., Altintas, I., and Berkley, C., Higgins, D., Jaeger, E., et al. "Scientific workflow management and the Kepler system." Concurrency and Computation: Practice and Experience. Vol 18, 2006.

[14] Gil, Y., Ratnakar, V., Kim, J., Gonzalez-Calero, P., Groth, P., Moody, J. and E. Deelman. "Wings: Intelligent Workflow-Based Design of Computational Experiments." IEEE Intelligent Systems, 26(1). 2011.

[15] Gil, Y. "Intelligent Workflow Systems and Provenance-Aware Software. Proceedings of the Seventh International Congress on Environmental Modeling and Software, 2014.

[16] De Roure, D., Goble, C. and R. Stevens. "The design and realizations of the myExperiment Virtual Research Environment for social sharing of workflows". Future Generation Computer Systems, 25 (561-567), 2009.

[17] Mates, P., Santos, E., Freire, J. and C.T. Silva. Crowdlabs: Social analysis and visualization for the sciences. 23rd International Conference on Scientific and Statistical Database Management, 2011.

[18] Peckham, S. D., Hutton, E.W.H. and Norris, B. "A component-based approach to integrated modeling in the geosciences: The design of CSDMS." Computers and Geosciences, 53, 2013.

[19] Shamir, L. Wallin, J.F., Allen, A., Berriman, B., Teuben, P., Nemiroff, R.J., Mink, J., Hanisch, R.J., K. DuPrie. "Practices in source code sharing in astrophysics." Astronomy and Computing, Vol 1, 2013.

[20] Zentner, L., Zentner, M., Farnsworth, V., et al. "nanoHUB.org: Experiences and Challenges in Software Sustainability for a Large Scientific Community." Journal of Open Research Software, 2(1), 2014.

[21] McLennan M. and R. Kennell. "HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering." Computing in Science & Engineering, 12(2), 2010.

[22] Patra, A., Jones, M., Gallo, S., et al "Role of Online Platforms, Communications and Workflows in Developing Sustainable Software for Science Communities." Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE2), 2014. figshare. http://dx.doi.org/10.6084/m9.figshare.1112569.

[23] Pérez, F. and B.E. Granger. IPython: A System for Interactive Scientific Computing, Computing in Science and Engineering 9(3), 2007.

[24] Torri, F., Clark, A.P. Dinov, I. Zamanyan, A. et al. Next generation sequence analysis and computational genomics using graphical pipeline workflows. Genes, 3:545-575. 2012.

[25] B. Giardine et al. Galaxy: A platform for interactive large-scale genome analysis. Genome Research, 15(10):1451-1455, 2005.

[26] Gil, Y.; Ratnakar, V.; and Garijo, D. OntoSoft: Capturing Scientific Software Metadata. In Proceedings of the Eighth ACM International Conference on Knowledge Capture, 2015.

[27] Barkley, J. Comparing simple role based access control models and access control lists. Proceedings of the Second ACM Workshop on Role-Based Access Control, 1997.