OntoSoft: Capturing Scientific Software Metadata

Yolanda Gil and Varun Ratnakar
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey CA, 90292
gil@isi.edu, varunr@isi.edu

Daniel Garijo
Ontology Engineering Group
Dpto. Inteligencia Artificial,
Facultad de Informática
Universidad Politécnica de Madrid
dgarijo@fi.upm.es

ABSTRACT
This paper presents OntoSoft, an ontology to describe metadata for scientific software. The ontology is designed considering how scientists would approach the reuse and sharing of software. This includes supporting a scientist to: 1) identify software, 2) understand and assess software, 3) execute software, 4) get support for the software, 5) do research with the software, and 6) update the software. The ontology is available in OWL and contains more than fifty terms. We are using OntoSoft to structure a software registry for geosciences, and to develop user interfaces to capture its metadata.

Author Keywords
Ontologies, software reuse, knowledge capture.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces - Interaction styles.

INTRODUCTION
Scientific software captures important knowledge and should be shared and reused. Although there are many popular code repositories used by scientists there is still a significant amount of software that is never shared. The reasons for not sharing scientific software include the desire to not expose less-than-ideal code, lack of incentives and credit for software, and interest in software commercialization among others [Howison and Herbsleb 2011]. While the loss of "dark data" in science is well recognized [Heidorn 2008], there is an analogous problem in the pervasive loss of "dark software".

Our interest is in supporting software sharing across the geosciences. With the exception of big model packages typically shared in modeling frameworks, geosciences software is rarely shared. In addition, it is scattered in different sites and not easy to find or reuse. Our goal is to improve software sharing by developing a software registry framework that includes metadata useful for discovery and reuse among scientists who are not software developers.

This paper introduces the OntoSoft software registry and its ontology for describing scientific software metadata. The ontology contains basic metadata properties to describe how to identify software, understand what it does and its utility for research, execute it, get support if questions arise, do research with it, and contribute to its development. These are all topics of interest to scientists, and the ontology revolves around those categories as a way to frame the requests for metadata in a practical light to incentivize scientists to provide it.

RELATED WORK
The Core Software Ontology (CSO) and the Core Ontology of Software Components (COSC) and [Oberle et al 2006] extend the DOLCE ontology [Gangemi et al 2002] to describe software components and web services. These ontologies were designed to describe large software systems, so their requirements include the 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. However, they focus on complex software systems, rather than in end users who are scientists and need to define software in terms of its reuse by other scientists.

Software repositories (e.g., GitHub, CRAN) are used widely by scientists. Although they allow users to describe their software, they do not use an ontology or model that can be exploited to support reuse.

REQUIREMENTS AND DESIGN OF ONTOSOFT
There are a few important requirements that we took into account in the design of OntoSoft.

First, the design of OntoSoft is centered on a broad range of users of a software registry for science. Although many scientists have sophisticated software development skills, the vast majority of scientists that should be able to share and reuse software do not. We considered it very important to design OntoSoft to be accessible to them.

K-CAP 2015, October 07-10, 2015, Palisades, NY, USA
© 2015 ACM. ISBN 978-1-4503-3849-3/15/10$15.00
DOI: http://dx.doi.org/10.1145/2815833.2816955

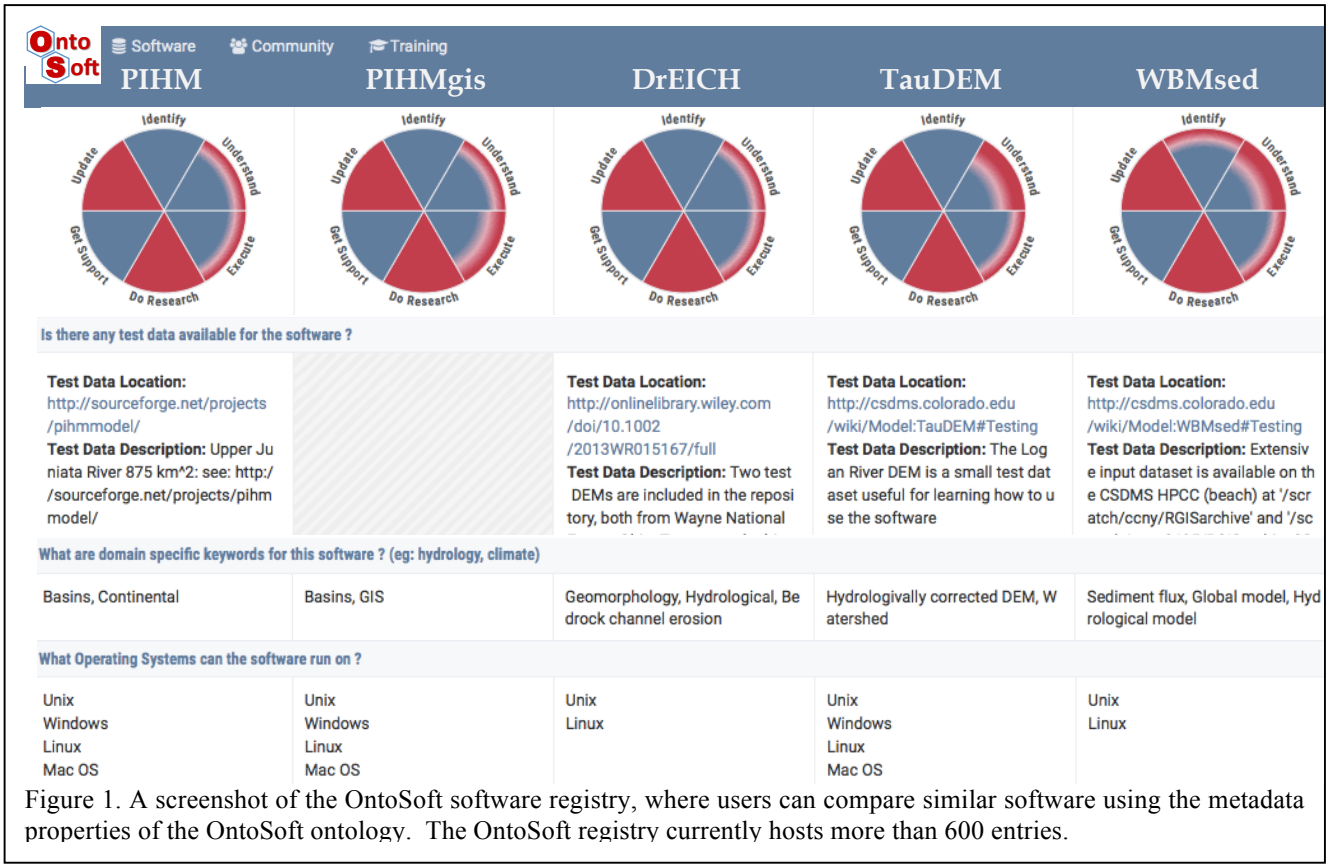

Figure 1. A screenshot of the OntoSoft software registry, where users can compare similar software using the metadata properties of the OntoSoft ontology. The OntoSoft registry currently hosts more than 600 entries.

Second, OntoSoft is intended for describing software written by scientists themselves. Although scientists use a wide range of software infrastructure to do their work, we do not intend for OntoSoft to be used to describe such infrastructure.

Third, the metadata captured by OntoSoft is focused on software sharing and reuse rather than software installation. The execution requirements for software can be quite complex and are often provided in detailed documentation. OntoSoft allows the specification of rather simple runtime requirements that scientists with limited software skills would want to specify about their software.

Fourth, scientists are never excited to provide metadata — not for datasets and not for software. We have created automatic extraction tools that get as much information as possible from existing software repositories. For example, if a user provides a pointer to GitHub, the system will extract license, implementation language, and other information automatically. In addition, an ontology that requires formal descriptions of every item might not have much uptake with end users. Therefore, OntoSoft provides high-level structure, but allows users to provide textual descriptions for most fields. We plan to develop automatic text extraction tools that can help users by structuring further the textual information that they provide. We envision a more formal ontology could be developed in the future, once the community is used to describing software and is more poised for adoption of a more formal ontology.

OVERVIEW OF ONTOSOFT

The metadata captured by OntoSoft falls into six major categories based on information that a scientist would seek about the software: 1) identify software, 2) understand and assess software, 3) execute software, 4) get support for the software, 5) do research with the software, and 6) update the software. The rationale for this organization is to allow the users who contribute software to understand why the metadata is requested, and to allow the users that are looking to reuse the software where to find metadata that is relevant to them.

Figure 1 shows a snapshot of the user interface that we developed with an earlier version of OntoSoft. It contrasts different metadata properties for several hydrological models.

The OntoSoft ontology and its documentation are available online[1]. It is implemented in OWL.

Figure 2 shows an overview of the major metadata properties to describe software in OntoSoft. We show in bold the major categories and subcategories of the properties. For each metadata property, we show its name

[1] http://www.ontosoft.org/

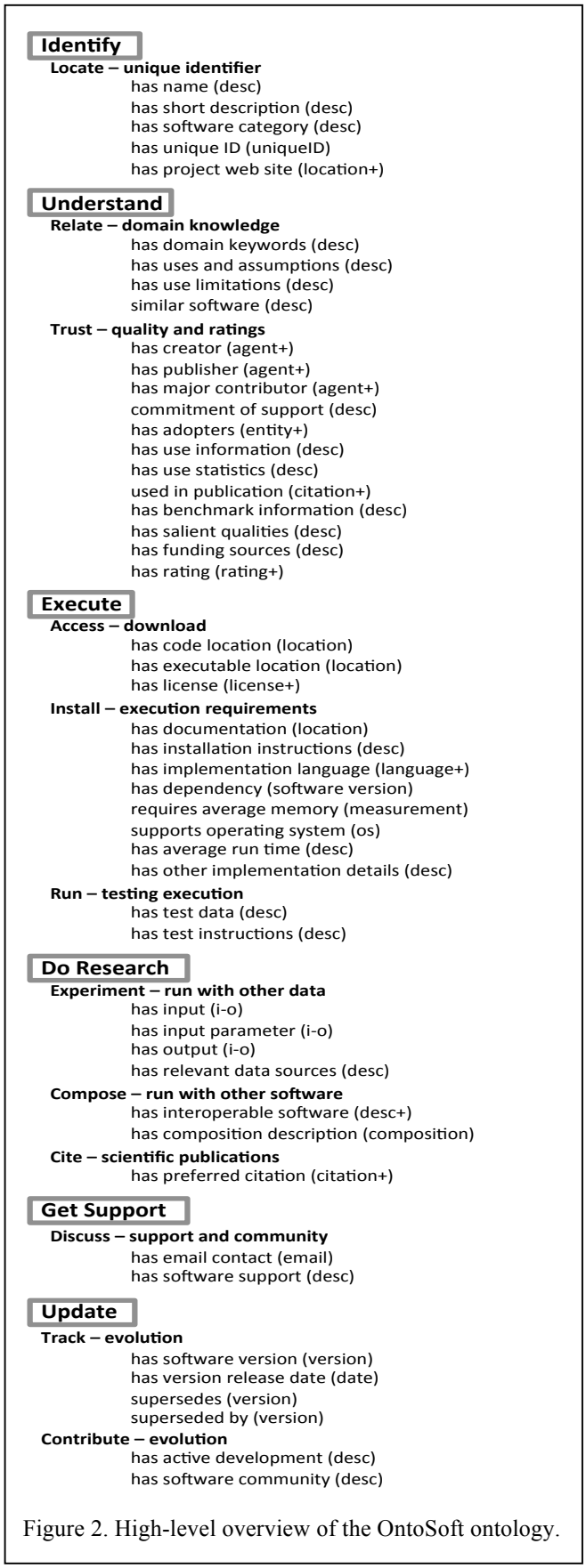

Figure 2. High-level overview of the OntoSoft ontology.

followed in parentheses by its range (i.e., the values that it can take) indicating with a plus sign when it can take more than one value. The remainder of this section gives an overview of the classes and properties of the OntoSoft ontology along the six major categories above. We note that many properties do not have structured values, instead their values are descriptions (in text). We believe that this will significantly reduce the burden placed on users to describe their software, as mentioned above.

Identifying Software

This category captures metadata that allows a user to identify a software entry. This can be, for example through the name of the software mentioned by a colleague, or through an identifier found in a paper.

These metadata properties include the name and a short description. The short description of the software provides important keywords to support search.

Another important property is a unique identifier. The unique identifier could be a DOI, a permanent URL, or a URI. Some software may have a unique identifier provided by a software repository (such as the identifiers provided by ASCL). For these reasons, the property takes a text value.

Understanding and Assessing Software

The metadata in this category allows users to understand what the software does, and to assess its utility for research.

Several metadata properties have to do with relating the software to knowledge about the domain. One way to find out details about what the software does in the context of science is to look up the web site for the software or for a related project, which is often associated with a software product. Other properties provide domain-specific descriptions of the software, including uses and assumptions, use limitations, and similar software that may be widely known in a community.

Another set of metadata properties have to do with trusting the software. One aspect of this is finding out who its creator and contributors were, as well as who published the software in the registry. This could be a person, a project, or an institution, so the value of these properties is an agent. Scientists have a name that carries a certain reputation, so the names would allow a user to check on the specific expertise of creators, contributors, and publishers. Another property allows users to specify the funding sources for the software. We also ask for reassurances on the commitment of support of the software, which could be whether it has a development community, whether the creator has been supporting it for several years, etc.

Another aspect of trust is understanding who is using the software and how they would rate it. The metadata properties to capture this include adopters (which again can be people, projects, entities), use statistics (extracted from the registry itself or cumulative download statistics that could be automatically extracted from the repository where

the software resides), and the publications that use the software.

Executing the Software

This category focuses on describing how to execute the software. This goes beyond instructions to install and run it, and includes how to access the software and how to run it with test data.

Several properties address the access of the code itself. They include license information, a location where the code resides (a repository or a local URL), and a location where a self-contained executable could be found. This is because many scientists do not want to go through the trouble of installing the original source code, and would prefer a pointer to an executable that can be directly run.

Other properties provide execution requirements that allow the installation of the software. These include documentation and installation instructions, the implementation language, the operating system, and average memory requirements. Another important property is the runtime dependencies, for example libraries that should be installed where the software is run. The average runtime is also important, and this would typically not be just a number but an explanation of what to expect the runtime to be depending on the size or characteristics of the data. We include a property to specify additional implementation details, described in free text form.

There are also properties that help a user run the software. These include pointers to test datasets, and instructions to check that the software runs properly and to check that the right results are obtained for the test data.

Getting Support for the Software

This category addresses how users can get support if they have any questions about the appropriate use of the software, have any problems with its installation, or wish to ask about specific cases not described in the documentation. This includes an email address to contact and a description of whether and how the software is supported.

Doing Research with the Software

This category includes metadata to enable scientists to do research with the software they want to use. Note that the metadata properties in the second category (understanding the software) are relevant here, but we expect the user is already aware of those since they would be considered before wishing to install and run the software.

Some metadata properties specify the particular input and output requirements of the software. This includes the types and constraints on the input data and parameters, as well as the expected outputs. Another important metadata property points the user to data sources where other data to run the software can be found.

Another set of properties is concerned with running the software in combination with other software,. This does not need to be specified as a formal workflow, it can be provided as a textual/diagrammatic description.

A metadata property is included to specify how to cite the software in a scientific publication.

Updating the Software

This category includes metadata to find new versions of the software, and to point users to a community that supports future extensions of the software. Some metadata properties are provided to track software version and release date, as well as properties to indicate whether there is a newer version and whether there are older versions that are superseded by the software. Finally, some metadata properties allow users to find whether the software is being actively developed, and pointers to on-line communities (mailing lists, issue tracking sites, etc.) who collaborate to further develop the software.

CONCLUSIONS

This paper describes the OntoSoft software registry designed using the OntoSoft ontology for describing metadata for scientific software. The ontology contains basic metadata properties to describe how to identify software, understand what it does and its utility for research, execute it, get support if questions arise, do research with it, and contribute to its development. A key contribution of OntoSoft is that its design is centered on users who are focused on doing scientific research, rather than software developers.

ACKNOWLEDGMENTS

We gratefully acknowledge the support from the US National Science Foundation with grant ICER-1440323. We would like to thank other members of the OntoSoft project, including Scott Peckham, Chris Mattmann, Erin Robinson, and Chris Duffy. We would also like to thank the many early adopters of OntoSoft, in particular Cedric David, Leslie Hsu, Anna Kelbert, and Sandra Villamizar.

REFERENCES

[1] Howison, J. and Herbsleb, J. D. “Scientific software production: incentives and collaboration.” Proceedings of the ACM Conference on Computer-Supported Collaborative Work (CSCW) , 2011.

[2] B. P. Heidorn. “Shedding Light on the Dark Data in the Long Tail of Science.” Library Trends, 57(2), 2008.

[3] 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.

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