An Ontology Design Pattern towards Preservation of Computational Experiments

Da Huo[1], Jarek Nabrzyski[1], and Charles F. Vardeman II[1]

University of Notre Dame
{dhuo,naber,cvardema}@nd.edu

Abstract. There has been work to provide preservation mechanisms that aim to capture the results of computational experiments. However, these preservation environments concern themselves with preserving the computational replication of numeric results rather than context of the calculation that is demanded for scientifically reproducible and extensible results. This is insufficient to help researches to comprehend preserved experiments and computational artifacts since sufficient documentation is lacking so understanding requires face to face meetings with their group members and external collaborators and may likely be lost. Computational experiments need to be described in both human and computer readable fashion. This paper aims at providing an ontology design pattern that conceptualizes any computational experiment artifacts and is aligned with other existing vocabularies such as the well-known W3C PROV vocabulary.

1 Introduction

As computational work becomes more and more integral to scientific research, an efficient approach to preserve and share experiments has become a concern required by the reproducibility of the scientific process. Although a variety of technologies have been created to capture computational experiments, it is still a struggle for scientists to understand the preserved experiment without a detailed documentation.

In this paper, we present an early work towards building an ontology called Smart Container (SC) that begins with conceptualizing computational experiments from the *perspective of computational environments and activities within those environments* as provided by the Docker Linux container framework[1] as a preservation tool. Docker is a lightweight virtualization platform that has several properties such as providing versioned file system, a modular design for distribution of software components as well as a sustainable community that make it attractive as a preservation tool. One community of collaborators at CERN[2] is already exploring Docker to preserve high energy physics experiments. Our approach is to eventually provide a mechanism that captures the additional provenance of computational experiments in a machine readable approach using the W3C standard RDF data model[3] that has been shown to aid in contextualization of scientific experiment.[6] We reference or align to existing ontologies

[1] http://docker.io
[2] https://twiki.cern.ch/twiki/bin/view/Main/DockerCVMFS
[3] http://www.w3.org/TR/2014/REC-rdf-schema-20140225/

and ontology patterns, such as PROV-O[4], CSO[5] and ACT[6] to aid in discoverability, interoperability and queryability and future extensibility.[4] By starting with a formal model of Docker and the provenance of Docker activities, we hope to provide a basis for 1) existing scientific workflow frameworks and their workflow descriptions to be captured *within* a Linux container environment 2) data to be integrated in a consistent manner and lastly 3) for a common description of the *environment*. The ultimate goal being to provide automated tools that "wrap" the existing Docker command line tool and infrastructure such that it is *transparent* to the scientist but captures information necessary to populate the metadata behind the scenes. Automated scientific gateways that utilize Docker as a deployment and execution platform would provide a further degree of transparency and allow researchers a low barrier for utilization.

2 Toward the Formalization of "Smart Containers"

We propose the construction of a "Smart Container Ontology" using a modular approach by **systematic** alignment of concepts present in Docker as a computational environment where computational activities occur by a modular approach reusing vocabulary terms where possible to contextualize those concepts and creation. This pattern based approach is believed to facilitate some control over unintended consequences of large ontologies and entailment inconsistencies introduced by their logic descriptions.[3] The modular ontologies PROV, CSO and ACT, three widely-used vocabularies. The purpose of connecting these patterns and creating a small specialization is to assist in answering competency questions such as 1) "What are the requirements for a computational activity?", 2)"What was the environment in which the activity was performed in terms of software components?",3)"What is the order in which provisioning activities must occur?", 4) "What software agents are responsible for a particular result or outcome". A brief description of these conceptual building blocks follows.

PROV-O is a W3C recommendation that describes "information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness"[7] and facilitates the exchange of this information between different systems and under different contexts. PROV-O has been demonstrated to have reasonable flexibility and has been shown to have promise for enhancing alignment between different ontologies.[2] Therefore, we propose to use PROV-O as a foundational building block to facilitate connection to other vocabularies and preservation efforts. The Core Software Ontology(CSO)[7] is a modular ontology formalizing common concepts from software engineering, such as data, software and execution of software on some hardware. We reference CSO using **rdfs:seeAlso** but do not import CSO directly to avoid entailment of relations beyond the scope of this application. ACT[8] ontology design pattern developed using the Vocamp

[4] `http://www.w3.org/ns/prov#`
[5] `http://cos.ontoware.org/cso#`
[6] `http://descartes-core.org/ontologies/activity/1.0/ActivityPattern#`
[7] `http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/`
[8] `http://ontologydesignpatterns.org/wiki/Submissions:An_Ontology_Design_Pattern_for_Activity_Reasoning`

process for ontology engineering[9], describes the common core concept for reasoning about activities.[1] It combines two perspectives for modeling activity, temporally-ordered entities in space-time and a workflow in planning-related applications. This axiomatization that facilitates temporal ordering of requirements and outputs essential to capture Docker as a provisioning environment is not provided by **prov:Activity** and must be imported from ACT.

2.1 A Proposed Pattern

A Docker based execution engine can be divided into at least two types of activities: **provisioning activities** that create an appropriate environment for a **computational activity** to occur that directly produce scientific observations. Provisioning activities include necessary procedures, such as prepare OS, install software, install software libraries, and download data, contributing to the "computational environment" for a scientific computation. Computational activities are actual experiment steps and workflows for a scientific investigation which involving generating and analyzing research data. We propose aligning these activity concepts with **prov:Activity** which represents "something that occurs over a period of time and acts upon or with entities.[5] Within Docker, an individual activity requires inputs to execute and it generates some result for future steps. A running container, its requirement is an image to start from and it generates a new image as an outcome. A software agent, such as bash, python and Docker, acting on behalf of a human, involves each activity is described by **prov:SoftwareAgent** and can connect to the human user as a **prov:Person**. Both concepts subclass **prov:Agent**. We propose identifying a human agent using URI's constructed from ORCID identifiers to facilitate investigator and potential publication identities to be propagated. Concepts referenced from CSO using **rdfs:seeAlso** include the human readable encoding of the whole computational experiment as **SoftwareAsCode**, and uses the pattern to seperate **InformationObject** from **InformationRealization** which is *executing* code from CSO.

In Fig.1. we present a concept map of the application of proposed pattern with respect to Docker. The class **sc:runningContainer**, representing a Docker container, is a specialization of **prov:Activity** and **act:Activity**. It has a requirement from ACT that is a **sc:RequiredImage**, a subclass of **prov:Entity** and **act:Requirement**. The outcome class from ACT is represented by **sc:OutcomeImage**, a subclass of **prov:Entity** and **act:Outcome**. A **sc:SoftwareAgent** is a direct subclass of **prov:SoftwareAgent** standing for the software executes activities on behalf of the human user. In Fig.2. we present a concept map of the specialization of the activity pattern. The classes **sc:provisonActivity** and **sc:computationalActivity** are a subclass of **sc:runningContainer**. Provisioning activities are influenced by **sc:provisionPlan** which subclass **prov:Plan**. A **sc:computationalActivity** is also influenced by a **sc:workflowPlan**[10]. Provisioning activities produce a computational environment that is a requirement for computational activities.

[9] `http://vocamp.org/wiki/Main_Page`

[10] see `http://www.w3.org/TR/2015/NOTE-hcls-dataset-20150514/#s7n_1` section 7.1.2 for details

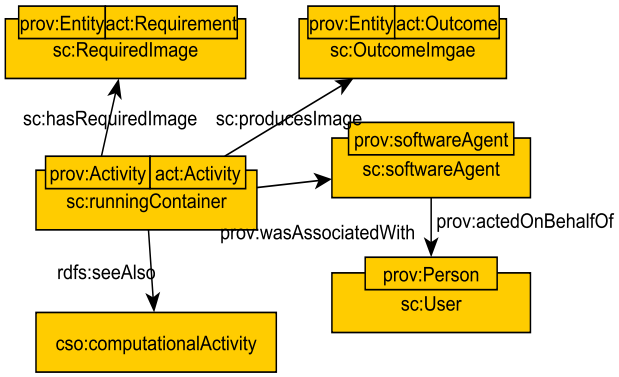

Fig. 1: Fragment of Docker Pattern

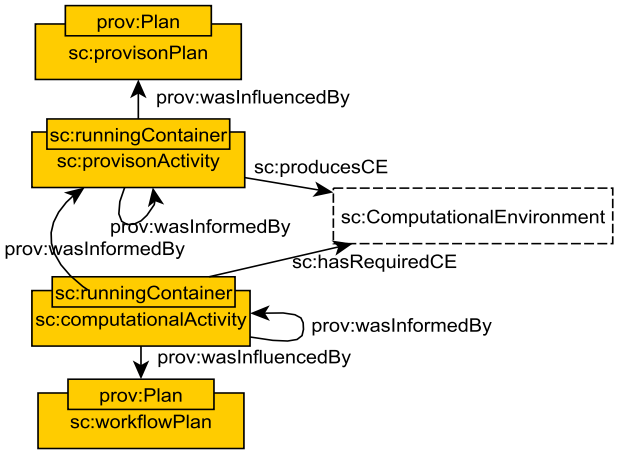

Fig. 2: Activity Pattern

3 Conclusion

In this paper, we present an ontology pattern named Smart Container that contextualizes computational experiments using Docker as an infrastructure example. We populated our ontology design pattern by analyzing main concepts in computational experiment and aligned with PROV-O, CSO and ACT to provide possibilities for wider extensions.

Acknowledgement We acknowledge funding from NSF grant PHY-1247316 "DASPOS: Data and Software Preservation for Open Science" and the Center for Research Computing at UND. We also acknowledge helpful conversations with Michelle Cheatham, Stian Soiland-Reyes, Matthew Gamble and Carole Goble.

References

1. Abdalla, A., Hu, Y., Carral, D., Li, N., Janowicz, K.: An ontology design pattern for activity reasoning
2. Compton, M., Corsar, D., Taylor, K.: Sensor data provenance: Ssno and prov-o together at last. In: To appear 7th International Semantic Sensor Networks Workshop (October 2014) (2014)
3. Gangemi, A.: Ontology design patterns for semantic web content. pp. 262–276. Springer (2005), `http://link.springer.com/chapter/10.1007/11574620\_21`
4. Janowicz, K., Hitzler, P., Adams, B., Kolas, D., Vardeman II, C.: Five stars of Linked Data vocabulary use. Semantic Web `http://iospress.metapress.com/index/053766UR810L7274.pdf`
5. Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J.: Prov-o: The prov ontology. W3C Recommendation 30 (2013)
6. Ma, X., Zheng, J.G., Goldstein, J.C., Zednik, S., Fu, L., Duggan, B., Aulenbach, S.M., West, P., Tilmes, C., Fox, P.: Ontology engineering in provenance enablement for the national climate assessment 61, 191–205, `http://linkinghub.elsevier.com/retrieve/pii/S1364815214002254`
7. Oberle, D., Grimm, S., Staab, S.: An ontology for software. In: Handbook on ontologies, pp. 383–402. Springer (2009)