Workflow-Driven Ontologies: An Earth Sciences Case Study

Leonardo Salayandia, Paulo Pinheiro da Silva, Ann Q. Gates, Flor Salcedo
The University of Texas at El Paso, Computer Science, El Paso, TX, 79905, USA
{leonardo, paulo, agates, fsalcedo}@utep.edu

Abstract

A goal of the Geosciences Network (GEON) is to develop cyber-infrastructure that will allow earth scientists to discover access, integrate and disseminate knowledge in distributed environments such as the Web, changing the way in which research is conducted. The earth sciences community has begun the complex task of creating ontologies to support this effort. A challenge is to coalesce the needs of the earth scientists, who wish to capture knowledge in a particular discipline through the ontology, with the need to leverage the knowledge to support technology that will facilitate computation, for example, by helping the composition of services. This paper describes an approach for defining workflow-driven ontologies that capture classes and relationships from domain experts and use that knowledge to support composition of services. To demonstrate the capability afforded by this type of ontology, the paper presents examples of workflow specifications generated from a workflow-driven ontology that has been defined for representing knowledge about gravity data.

1 Motivation

The NSF-funded Geosciences Network (GEON) project is a collaborative effort among researchers from a broad cross section of computer science and earth science disciplines [1]. GEON is comprised of computation clusters that include compute nodes and data nodes that store high quality geological information and software services and that enable data access, analysis, modeling, and visualization. GEON is an example of a "virtual scientific community" that uses cyber-infrastructure (CI) to support and enhance the scientific process.

Different groups from the GEON community are developing ontologies that support search [2], information integration [3], and service discovery [4]. An ontology [5] is an explicit specification of the objects, concepts, and other entities that are assumed to exist in a specific domain and the relationships that hold among them.

In addition to development of ontologies, many research teams are using workflow techniques to specify the computation of complex scientific activities [6,7,8]. In this paper, we describe a new approach to ontology design called *Workflow-Driven Ontologies (WDO).* WDOs are distinguished from *domain-based ontologies* that capture basic knowledge about a domain. Use cases typically drive the specification of domain-based ontologies [9]. In the WDO approach, abstract workflow specifications drive the elicitation and specification of classes and their relationships. For example, domain experts, i.e., earth scientists, begin the knowledge acquisition process by identifying a *product* and from the *product* identify *methods* that can generate the product. Further, domain experts can identify *data* that are required as input for the identified methods. Knowledge acquisition methodologies based on WDOs are flexible since earth scientists can refine WDOs by refining a WDO-derived workflows and vice-versa. We claim that abstract WDO-derived workflow specifications are indeed the use cases for WDOs.

Prior to presenting the details of WDOs, this paper first motivates the utility of WDOs by presenting a case study in Section 2. The case study illustrates how a contour map can be generated from composition of services from the gravity domain. Section 3 explains how workflows are derived from WDOs using the WDO class hierarchy and core relationships. Section 4 presents related work including a discussion on how WDOs compare to other ontologies. Section 5 summarizes the main contributions and open issues related to the development of the WDO approach.

2 Gravity Case Study

This section describes the Gravity ontology [10] and the use of the ontology to specify workflows. To remain consistent with the terminology used by the OWL Web Ontology Language community, we use the term "class" to denote types of objects captured by the ontology.

2.1 The Gravity Contour Map Scenario

The Gravity ontology specifies several scientific products, e.g., contour maps and anomaly maps. In addition to products, the Gravity ontology specifies other classes and relationships related to scientific workflows such as data sets and methods. Products may be derived from different methods. As a result, numerous workflows may be derived from the ontology for each product. There are multiple ways a user or application can use the ontology to support the generation of a complex result, i.e., the composition of methods generating a given product. Consider a simple scenario in which an earth scientist wishes to acquire a contour map of gravity data. In this scenario, an earth scientist accesses the portal, outlines a footprint for the area of interest, and requests the appropriate map. In spite of the simplicity of this request, there are a number of possible ways for an application to use GEON resources to generate a response, and each way the CI is used presents new challenges. The following variations may occur:

1. Many maps are stored in several servers, and only one map matches the request. The portal presents that map to the user.
2. More than one map matches the request, and the portal presents a list of the maps with a trust recommendation for each map. The trust level for each map is computed based on the user's degree of trust on sources, source authors and other users. The user selects the desired map.
3. More than one map matches the request, and the portal displays the map with the highest trust level for the user.
4. The map does not exist for the footprint specified; the GEON CI identifies a composition of services required to construct the map, i.e., starting from data access and retrieval, data filtering services, and services to render the map. In the case where there are several service alternatives, e.g., several gravity data sources, the GEON CI, either automatically or with user interaction, filters the services to use in the composition based on computed trust levels for the user.

The next two subsections elaborate Scenario 4, i.e., automation of the composition of services to create a Gravity Contour Map with the support of an ontology.

2.2 The Gravity Ontology

Figure 1 presents a visual representation of three class hierarchies from the Gravity ontology and the classes related to producing a gravity contour map. The Gravity ontology specifies multiple relationships between classes across the three hierarchies; for clarity the relations that are associated with the classes are listed in the sidebar of the figure rather than shown graphically.

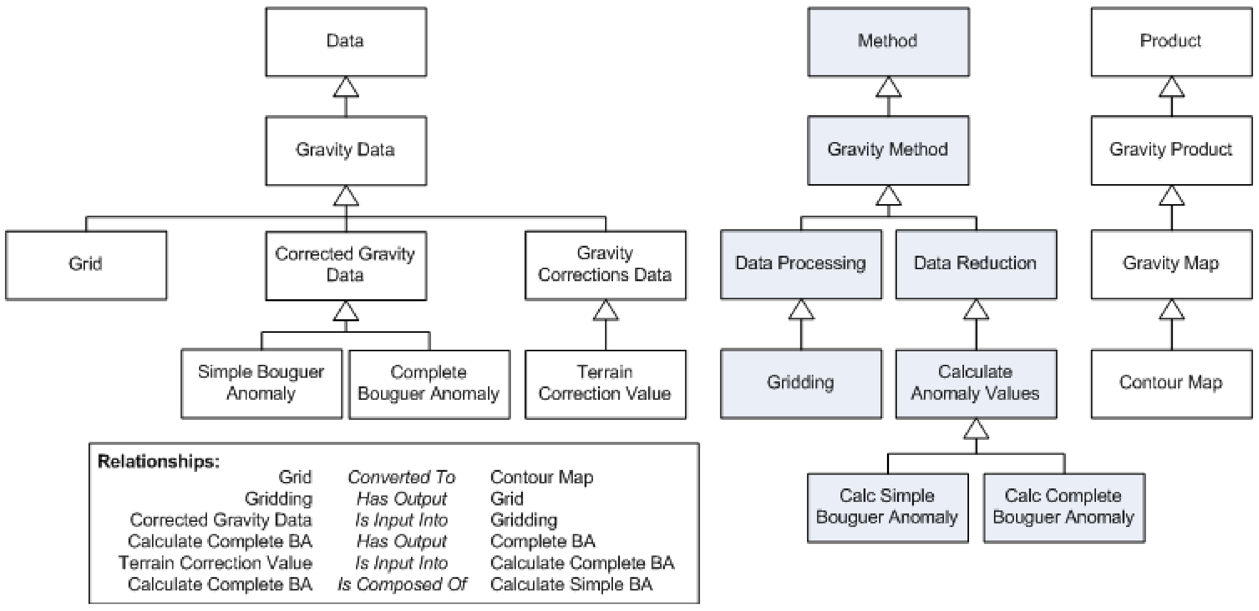
It is expected that the different types of CI services are represented as classes defined in an ontology used to create executable workflows. Consequently, CI services that correspond to classes under the *Data* hierarchy of the ontology are services that provide access to data repositories; CI services that correspond to classes under the *Method* hierarchy are services that take data as input, provide some functionality that can transform the data, and outputs the transformed data; and CI services that correspond to classes under the *Product* hierarchy are services that provide access to an artifact library.

The relationships between classes provide the basic roadmap to specify complex CI functionality through composition of services. As an example, consider the first row of the relationship sidebar in Figure 1 that shows the *Converted To* relationship between the classes *Grid* and *Contour Map*. This relationship suggests that, given a service that corresponds to a Grid class, a service composition is viable that would result in a *Product* artifact corresponding to a *Contour Map* class.

2.3 Contour Map Workflows

This section describes two workflows that generate gravity data contour maps. The workflows are derived from the classes and relationships specified in the Gravity ontology. The first workflow, shown in Figure 2, creates a contour map from *Simple Bouguer Anomaly* gravity data. The second workflow, shown in Figure 3, extends the workflow of Figure 2 to produce a contour map from *Complete Bouguer Anomaly* gravity data.

The workflow in Figure 2 is divided into two main sections. The left-hand side represents the classes of type *Information* that are associated with the workflow, and the right-hand side represents the classes of type *Method* that are involved in the transformation of the information required to achieve the desired outcome, i.e., a contour map. The left-hand side of the diagram is divided further into two sections: *Product* and *Processed Data.* The distinction between these classes is explained in Section 3.



Grid	*Converted To*	Contour Map
Gridding	*Has Output*	Grid
Corrected Gravity Data	*Is Input Into*	Gridding
Calculate Complete BA	*Has Output*	Complete BA
Terrain Correction Value	*Is Input Into*	Calculate Complete BA
Calculate Complete BA	*Is Composed Of*	Calculate Simple BA

Fig. 1: Hierarchies from the Gravity ontology.

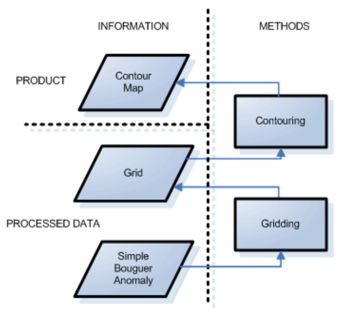


Fig. 2: Workflow specification to produce a contour map from *Simple Bouguer Anomaly* data.

The *Simple Bouguer Anomaly Contour Map* workflow shown in Figure 2 produces a *Contour Map* product that is output from the *Contouring* method. The *Contouring* method takes *Grid* processed data as input. The interaction between these classes in the workflow is realized by the *Grid* and *Contour Map* data classes and the *Converted To* relationship specified between them (cf. Figure 1); however, in order to make the relationship between these two classes more appropriate for workflow specification, there must be a new class of type *Method*, i.e., *Contouring*, and associated relationships to signify that *Contouring* takes *Grid* and transforms it into *Contour Map*. Since the original draft of the Gravity ontology does not include the *Contouring* method, the scientist is cued to extend the ontology by adding an intermediary method class to the ontology.

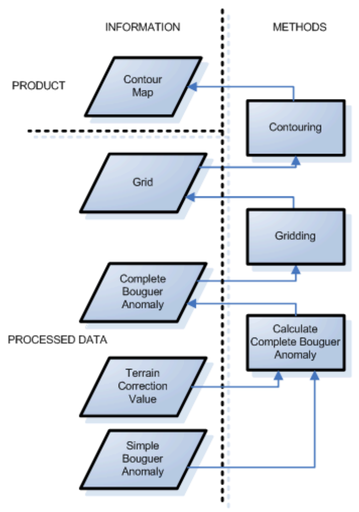


Fig. 3: Workflow specification to produce a contour map from *Complete Bouguer Anomaly* data.

Following the workflow specification of Figure 2, in the case of *Gridding* and *Grid*, the relationship *Has Output* makes the workflow specification straightforward— no modification to the Gravity ontology is necessary. For the case of *Gridding* and *Simple Bouguer Anomaly*, the Gravity ontology does not have a direct relationship, but it does include a relationship between the classes *Corrected Gravity Data* and *Gridding*. Since *Corrected Gravity Data* is a parent class to the *Simple Bouguer Anomaly* data, the gap can automatically be closed with the aid of a software reasoner that understands the basic hierarchical relationship between classes.

The workflow for *Complete Bouguer Anomaly* gravity data, shown in Figure 3, has a structure similar to the *Simple Bouguer Anomaly* workflow, but with additional steps to produce the *Complete Bouguer Anomaly* data.

3 Workflow-Driven Ontologies (WDOs)

The notion of Workflow-Driven Ontologies (WDOs) stem from efforts to produce a domain-based ontology by a February 2004 Seismology Ontology workshop held at Scripps Institution of Oceanography in San Diego, California. The attendees of the workshop included experts in the areas of seismology and information technology.[1] The result was a categorization and relationship model that supports workflow specifications. These categories drive the classes that are to be elicited from scientists in defining a workflow-driven ontology, and the relationships between the classes that form the foundation to produce workflow sequences that can be mapped to CI development efforts.

The following sections discuss characteristics that differentiate a WDO from a domain ontology, specifically the class categorization and relationships of WDOs and a methodology used to produce workflow specifications from a WDO. This section also presents a WDO software API and a WDO assistant that complements the methodology.

3.1 WDO Classes

A basic workflow specification can be considered the application of a method that takes information as input and yields information as output (information → method → information). Figures 2 and 3 denote compound workflow specifications in which information is submitted as input to a method and transformed and this basic workflow pattern repeated until the desired information is derived.

Figure 4 shows the class hierarchy that forms the basis for a WDO. Notice that WDOs are OWL ontologies that always present the classes (also referred to as concepts) as shown in Figure 4. As OWL ontologies, the class hierarchies of WDOs are grounded in the OWL class *Thing*. The class hierarchy is a result of our initial Gravity WDO prototype, which is implemented in OWL, and interactions with experts in the field of geophysics.

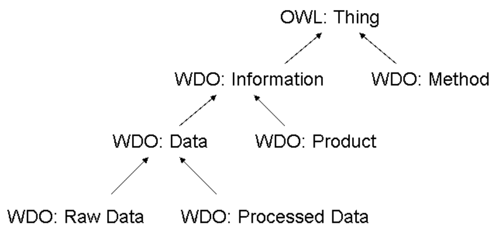


Fig. 4: Class hierarchy of a WDO.

To be consistent with the notion of a basic workflow specification and to facilitate the production of workflow specifications from ontologies, the first requirement to create a WDO is to categorize classes into *Information* and *Method*. The gravity WDO specializes *Information* into *Data* and *Product*. The justification for this distinction is that classes categorized as *Data* are considered first-class citizens in CI-related scientific activities while classes categorized as *Product* are considered artifacts that can be reproduced given a reliable data source. It is the authors' belief that this categorization of *Information* is applicable to other scientific fields employing CI and should be preserved as a general requirement for WDOs.

The *Data* class is further broken down into *Raw Data* and *Processed Data*. *Raw Data* is what is referred to as "measured" data or data that is in its natural form, i.e., data that has not been transformed through the application of some *Method*. On the other hand, classes categorized as *Processed Data* represent data that has undergone a transformation from an initial state through the application of some *Method*. This distinction is done in order to support basic rules for workflow construction, where we can identify when a workflow construction process should stop because it has reached the "base" class of information, i.e., *Raw Data*. The capture of provenance information [11] would also benefit from this distinction by providing the scientist with the ability to annotate Raw Data with source metadata, e.g., sensor metadata, and to annotate Processed Data with method metadata, e.g., metadata about the method generating the data.

An important feature about the separation of the workflow classes into different levels of *Information* is that *Product* and *Processed Data* can be input or

[1] Randy Keller and Ann Gates, University of Texas at El Paso; Bertram Ludaescher, Dogan Seber, Chaitan Baru, and Kai Lin, San Diego Supercomputer Center; Gabi Laske and Frank Vernon, Scripps Institute, University of California at San Diego; Tim Ahern, IRIS; Colin Zelt, Rice; Matt Fouch, Arizona State; John Hole, Virginia Tech; David James, Carnegie Institute of Washington; Bill Pike, Penn State

output to classes categorized as *Method*. For example, the end product of the workflows depicted in Figures 2 and 3, i.e., *Contour Map*, could be used as input to another method to produce a more complex product. On the other end of the workflow, *Processed Data* can be considered output of another method that takes additional data as input, and this workflow refinement could be iterated until a class categorized as *Raw Data* is reached.

Method classes in the WDO have a unique signature for input and output information. This requirement simplifies the process of automating workflow generation by uniquely identifying the *Information* classes related to a given *Method* on a workflow specification.

3.2 WDO Relationships

The TAMBIS Ontology [12] that uses relationships to link functions influenced the WDO approach of defining relationships between classes denoting functional objects and other types of classes. The relationships presented in Figure 1 demonstrate how classes are linked to one another in a WDO. Figure 5 illustrates how these relationships generalize to link *Information* and *Method* classes in a workflow specification.

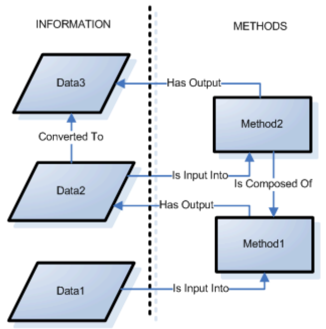


Fig. 5: Shortcut and core workflow relationships.

Because scientists specifying WDO classes and relationships may not think in terms of a basic workflow specification, i.e., *Information* → *Method* → *Information*, scientists may under specify a WDO. For example, classes categorized as *Information* may be connected without an intermediary class of type *Method*. Consider the Gravity WDO presented in Figure 1 that shows the *Converted To* relationship between *Grid* and *Contour Map*. Similarly, scientists may also relate *Method* classes without intermediary classes of type *Information*. For example, the Gravity WDO of Figure 1 shows method classes *Calculate Complete Bouguer Anomaly* and *Calculate Simple Bouguer Anomaly* and the relationship *Is Composed Of* that links them. In WDO terms, these types of relationships, which result in an underspecified WDO, are referred to as *shortcut relationships*. Shortcut relationships are defined between *Information–Information* and *Method–Method* classes, and they indicate the need for an intermediary class and associated relationships to produce a valid workflow sequence. To illustrate this, Figure 5 shows *Data2* connected to *Data3* through the *Converted To* relationship. In order to make the WDO consistent with a workflow specification, the WDO has to have some method, *Method2* in this case, to relate *Data2* and *Data3*. As mentioned above, since WDO *Method* classes have a unique signature to specify input and output, two *Information* classes can only be linked in a single step by a unique method.

It is expected that the shortcut relationships in a WDO will vary depending on the domain that the WDO addresses. For example, the Gravity WDO contains a shortcut relationship *Converted To*, while a WDO related to biology may have a shortcut relationship named *Metamorphosed To*.

Finally, the relationships *Is Input Into*, and *Has Output*, as well as their corresponding inverses *Has Input* and *Is Output Of* are referred to as *core workflow relationships* and are required in a WDO to build the workflow specification patterns between the ontology classes. Core workflow relationships should appear in every WDO with the same name, regardless of the domain that the WDO addresses.

3.3 Generating Workflows from WDOs

Similar to software agents, the process of generating workflows from WDOs is a task-oriented activity where the scientist starts the process by specifying the desired end result. End results would typically be of type *Information*, which in the case of WDOs is further broken down to *Raw Data*, *Processed Data*, and *Product*. In the case of *Raw Data*, the WDO would typically be a one-step workflow specification in which data is retrieved from a data source. In the case of either a *Product* or *Processed Data*, the solution to the workflow would be composed of inputting *Information* to a *Method* and getting the desired *Information* as output from that *Method*. The *Information* → *Method* → *Information* pattern is repeated until the user is comfortable with the level of granularity and/or availability of input information to achieve the desired output information.

The *Information* → *Method* → *Information* patterns are composed based on the classes and relationships specified in the WDO. However, given that the WDO may be developed by the scientific community initially as a domain ontology, there may be situations where dead-ends are reached and the scientist would then have to complete the workflow specification by adding new classes. In such situations, by inspecting workflow specifications derived from a WDO, the scientist would provide feedback to the WDO to add new relationships and/or introduce new classes between existing classes that would produce a path to complete the workflow specification.

Three basic rules have been defined that provide initial guidelines for specifying workflows:

1. For every class of type *Product* and *Processed Data* included in the WDO, there must be at least one relation with a *Method* class that specifies that a class of type *Product* or *Processed Data* is the output of the given *Method* class.
2. For every class of type *Method* included in the WDO, there must be at least one relation that specifies that a class of type *Information* is the input to the given *Method* class.
3. For every class of type *Method* included in the WDO, there must be at least one relation that specifies that a class of type *Information* is the output to the given *Method* class.

Furthermore, existing shortcut relationships in the WDO may serve as additional guidelines to elicit new workflow relationships as discussed in the previous section.

With respect to the introduction of new classes into the WDO, it is important to notice that the ontology refinements should avoid any changes to the WDO upper-level classes; otherwise, there may be non-intended side effects that can cause the WDO to become unstable, possibly producing erroneous workflow specifications or introducing unreachable classes. Other work from Parsia et al. [13] that deals with OWL ontology debugging may be complementary to the WDO refinement process.

Details about methodologies for ontology refinement are out of the scope of this paper. We expect, however, that scientists will be able to transform publicly available ontologies into WDOs and from these new ontologies start developing workflow specifications that satisfy his or her specific needs.

3.4 The WDO API and Assistant

A WDO API has been prototyped that facilitates the integration of WDOs into toolkits from different domains. The WDO API is built on top of the Jena2 Ontology API [14] that provides functionality to access OWL ontologies through Java programming. The WDO API offers specific methods that facilitate the WDO refinement process, as well as functionality to create workflow specifications as discussed in the previous section. The WDO API provides the following functionality:

- List, Add, and Edit WDO classes by category, e.g., classes that are classes of *Raw Data*, *Processed Data, Product*, or *Method*;
- Search for workflow sequence patterns for a given class of type *Information*;
- Identify missing core workflow relationships between *Information* and *Method* classes and suggest intermediary classes based on existing shortcut relationships.

In order to provide end-users with a useful tool to create WDOs, the WDO Assistant has been prototyped as a stand-alone application based on the WDO API that provides a GUI to assist scientists to create new WDOs, as well as to extend domain ontologies into WDOs. In addition, the WDO Assistant allows scientists to generate workflow specifications for selected *Information* classes and to provide feedback to the scientist when the generated workflows are underspecified. The WDO Assistant is divided into the following main interaction modes:

- Brainstorm Mode: Allows scientists to create new WDO classes or enhance existing domain ontology classes to align with the WDO class hierarchy;
- Elicitator Mode: Allows scientists to create, edit, and remove workflow relationships between classes;
- Workflow Generator Mode: Allows scientists to chose an *Information* class and generate workflow specifications to produce it.

The initial prototype of the Workflow Assistant will produce executable workflows using OWL-S [15] and MoML [16]. OWL-S is a markup language that facilitates the automation of web services tasks, and MoML is the modeling markup language used by the Kepler Scientific Workflow Engine [7]. In order to create an executable workflow, a mapping has to be established between the WDO classes involved in the workflow and the CI resources that carry out the actual work, e.g., web services. Our initial prototype assumes that this mapping is manually established by the user; however, mechanisms like OWL-S allow for more dynamic settings where the workflow engine can search through the CI to look for matching resources based on some semantic description.

4 Discussion and Related Work

Ontology development for the sciences is a community-driven process where consensus is needed from domain experts on issues that may be on the cutting edge of research. As a result, it is typically hard to validate scientific ontologies. Workflow specifications generated from a WDO can be used by scientists to validate a WDO. For example, by examining the workflow produced, a scientist can decide whether the relationships specified between classes in the WDO are correct or need to be modified. By providing this kind of feedback to the WDO, the scientist can refine the WDO, and ultimately measure the level of correctness of the WDO relative to the workflows produced by the WDO. Furthermore, it is expected that as workflow specifications are created from the WDO, not all ontology classes will be available as corresponding CI resources. The WDO can serve as a roadmap towards CI development by allowing scientists or software developers to decide which CI resources to implement in order to realize a potentially useful workflow, as well as to indicate what kind of parameters such resources must have as input and output.

Related work includes the TAMBIS ontology [12], which similar to WDOs, uses advanced categorization of concepts and relationships. TAMBIS is a bioinformatics ontology whose design is based on description logics in order to allow dynamic creation and reasoning about the concepts. In a similar way, WDOs also use OWL inference engines to reason about their concepts, e.g., to infer inherited relationships through super-classes. The TAMBIS ontology recognizes the importance of distinguishing between various representations of a concept and, therefore, it is organized into multilayer divisions. For example, in the bioinformatics world, a structure can be separated into its physical and abstract representations. Thus, the *Generalized Structure* division for a concept is separated into *Physical Structure* and *Abstract Structure.* Also, the ontology has separate concept divisions for *biological processes* and *biological functions*. This notion of distinguishing between the possible representations of a concept helps reinforce the idea that separating concepts into categorizations is beneficial. The separation of different concerns regarding concepts for a domain knowledge was adopted in the Gravity WDO in order to separate actual data from the actions and results of the use of data in the ontology, i.e., from the *Method* and *Product*.

The Gene Ontology (GO) [17] is a controlled vocabulary about gene information. It is split up into three main categories, the cellular component ontology, molecular function ontology, and the biological process ontology. In the GO ontology, a function is similar to a method in the gravity ontology and a process is a series of steps, which is similar to a workflow in the Gravity WDO.

The Semantic Web for Earth and Environmental Terminology (SWEET) ontologies [18] were developed to capture knowledge about Earth System science. A group of scientists have been capturing several thousand Earth System science terms using the OWL ontology language. There are two main types of ontologies in SWEET: facet and unifier ontologies. Facet ontologies deal with a particular area of Earth System science (earth realm, non-living substances, living substances, physical processes, physical properties, units, time, space, numerics, and data). Unifier ontologies were created to piece together and create relationships that exist among the facet ontologies. Facet ontologies use a hierarchical methodology in which children are specializations of their parent nodes. The SWEET ontologies are currently being used in GEON to capture geologic processes and terms.

5 Conclusion

This paper presents the workflow-driven ontology (WDO) approach. With the introduction of an upper-level class hierarchy of workflow-related classes, WDOs facilitate the process of creating ontologies to be used on scientific domains. With the introduction of a well-defined set of relationships between classes, WDOs are used to guide scientists through the process of relating classes in a way that can later be polished into useful workflow specifications. With the introduction of shortcuts, WDOs enable knowledge capture for underspecified processes that can be refined later.

The WDO approach has been informally developed and used in GEON during the last four years; however, just recently the approach has been formalized and WDO-specific tools have been developed to replace generic tools such as OWL editors and spreadsheets. This work is initially focused on the Gravity WDO, which stems from an ontology on gravity data and its application to Geophysics. Next steps include applying the WDO approach to other scientific fields that are nascent in the application of CI. The WDO approach has demonstrated to be useful to integrate the efforts of end-users, e.g., scientists, and computer scientists to relate and discuss technical details of CI implementation. The WDO tools and methodologies are less mature than the WDO approach itself. Our future work includes further evaluation about the usefulness of our WDO tools.

6 Acknowledgements

This work is funded in part by the National Science Foundation (NSF) through grant NSF EAR-0225670. The authors would like to thank Dr. Randy Keller for his vital contributions in developing the Gravity WDO, and Dr. R. Hannes Niedner from SDSC for his valuable feedback.

7 References

[1] The Geosciences Network: Building Cyberinfrastructure for the Geosciences, http://www.geongrid.org/, May 2006.

[2] L. Ding, et al., "Swoogle: A Search and Metadata Engine for the Semantic Web", *Proc. 13th ACM Conf. on Information and Knowledge Management*, November 2004.

[3] A. Doan, A.Y. Halevy, and N.F. Noy, "Semantic Integration Workshop at the Second International Semantic Web Conference (ISWC2003)", *SIGMOD Record*, 33(1), 2004, pp. 138-140.

[4] D.J. Mandell, and S. McIlraith, "Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation", *Second Intl. Semantic Web Conference (ISWC2003)*, October 2003.

[5] T. R. Gruber, "A translation approach to portable ontologies", *Knowledge Acquisition*, 5(2):199-220, 1993

[6] M.N. Alpdemir, et al., "Contextualised Workflow Execution in myGrid", *Proc. European Grid Conference,* Springer-Verlag LNCS 3470 2005, Science Park Amsterdam, The Netherlands, February 14-16 2005, pp. 444-453.

[7] B. Ludäscher, et al., "Scientific Workflow Management and the Kepler System", *Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows*, 2005.

[8] N. Alpdemir, et al., "OGSA-DQP: A Grid Service for Distributed Querying on the Grid", *Proc. 9th EDBT Conference*, 2004, pp. 858-861.

[9] N.F. Noy, and D.L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology". *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05*, March 2001.

[10] F. Salcedo, "A Method for Designing Computation-Driven Ontologies in the Geosciences", *Master's Thesis*, The University of Texas at El Paso, May 2006.

[11] P. Pinheiro da Silva et al. "Knowledge Provenance Infrastructure". *IEEE Data Engineering Bulletin*. Vol. 26 No. 4, pages 26-32, December 2003.

[12] P.G. Baker, et al., "An Ontology for Bioinformatics Applications," *Bioinformatics,* 15(6), 1999, pp. 510-520.

[13] B. Parsia, E. Sirin, and A. Kalyanpur, "Debugging OWL Ontologies", *Proc 14th Intl Conference on World Wide Web*, Chiba, Japan, 2005, pp. 633-640.

[14] The Jena2 Ontology API, http://jena.sourceforge.net/ontology/index.html, July, 2006.

[15] "OWL-S: Semantic Markup for Web Services", The OWL Services Coalition, December, 2003.

[16] E. Lee, and S. Neuendorffer, "MoML – A Modeling Markup Language in XML – Version 0.4", Technical Memorandum ERL/UCB M 00/12, University of California at Berkley, March 14, 2000.

[17] R. Allen, and L. Fisher, ed., Workflow: An Introduction, Future Strategies, Lighthouse Point, FL, 2001, pp. 15-38.

[18] Guide to SWEET ontologies, http://sweet.jpl.nasa.gov/guide.doc, April , 2006.