iPlant SSWAP (Simple Semantic Web Architecture and Protocol) enables semantic pipelines for biodiversity

Damian D. G. Gessler[1], Blazej Bulka[2] , Evren Sirin[2], Hans Vasquez-Gross[3], John Yu[3], Jill Wegrzyn[3]

[1]The iPlant Collaborative, University of Arizona, Tucson, AZ, U.S.A.
dgessler@iplantcollaborative.org
[2]Clark and Parsia, Washington, D.C., U.S.A.
{blazej,evren}@clarkparsia.com
[3]University of California, Davis, CA, U.S.A.
{havasquezgross,jjsyu,jlwegrzyn}@ucdavis.edu

Abstract. Real-time response is a basic characteristic of the Web. Yet semantic reasoning at transaction-time supporting real-time response remains challenging. Here we report how the iPlant Semantic Web Platform uses SSWAP (Simple Semantic Web Architecture and Protocol; http://sswap.info) for transaction-time reasoning, service discovery, workflow construction, and execution. The platform enables users at web sites, such as TreeGenes' DiversiTree and CartograTree, to select data and use it for real-time semantic discovery into a knowledge base of semantic web services. The platform uses first-order, description logic reasoning and just-in-time ontologies to allow users to drag-*n*-drop independent, distributed semantic web services into a semantic pipeline. This enables biodiversity research using data sets from TreeGenes, FLUXNET (Ameriflux), WorldClim, and TRY-DB integrated under a common web front-end called CartograTree. Scientific use cases are for tree scientists to associate phenotype and/or environmental traits with underlying genotypes in geo-referenced forest trees across a distribution of Web resources.

Keywords: Semantic Web Services, SSWAP, iPlant, TreeGenes, DiversiTree, CartograTree, OWL

1 Introduction

Bioinformatic software exhibits long-tail characteristics: a relatively small number of programs and web sites are widely used (*e.g.*, [1,2]), while a much larger number are used by varied audiences for specialized applications (*e.g.*, [3,4]). The iPlant Collaborative [5] seeks to enable data-driven scientific integration, both within the enterprise and across Web resources, including widely used programs of general interest and niche programs for specific needs. Emphasis is on having software layers handle data and service syntax and semantics (including independently developed and maintained long-tail offerings) thereby freeing the scientist to focus on data and service discretionary use. To achieve this, iPlant is using SSWAP (Simple Semantic Web Architecture and Protocol [6]) in a drag-*n*-drop semantic pipeline motif with third-party Web site integration. In this paper we report how a collaboration with TreeGenes [7] enables biodiversity applications in forest genetics. Exemplary applications in land management and biodiversity include the identification of specific genotypes that may be best suited for reforestation, or the development of strategies for tree migration as it relates to climate change. In both cases, genotypes that influence traits such as cold-hardiness, drought-tolerance, and disease resistance can be examined in relation to environmental characteristics of target regions including elevation, soil composition, and precipitation.

2 The Platform

Architecture The iPlant Semantic Web Platform is a Web architecture of four distributed actors: *i*) providers of services; *ii*) consumers of services; *iii*) ontology severs; and *iv*) a semantic Discovery Server (pipeline-maker and match-maker). Data—be it unstructured, semi-structured, or structured (*e.g.*, as in relational database stores)—enters the system via a service interface layer; *i.e.*, the platform does not operate on raw data *per se*, but via service interfaces, the invocations of which yield access to, and transformations of, data. This service interface layer is key to enabling distributed data to be integrated "rationally" under a first-order description logic protocol.

SSWAP (Simple Semantic Web Architecture and Protocol) SSWAP is a 100% W3C OWL DL-compliant light-weight protocol of five classes and 12 properties. It allows services to describe what they are, the types of data they consume, and the types of data they produce. The protocol's ontology in its entirety is at [8]. The five classes correspond to: *i*) the service Provider, *ii*) the service itself, called a Resource, *iii*) a data structure construct called a Graph, *iv*) input data (a Subject), and *v*) output data (an Object). SSWAP is the service analog of the fundamental RDF data model of mapping a subject to an object via a property; in the cases of SSWAP, the protocol maps a Subject to an Object via the implicit operation of a service (the Resource). Subject and Object instances may be URIs, thereby allowing for indirection and non-serialization of data, or they may identify data structures of arbitrary OWL sub-graphs, with properties and serialized data. Instances of Resource, Subject, and Object may be annotated with user-defined ontologies and thus are "unlimited" in domain scope; the protocol simply defines the scaffold. Services may have multiple Subjects mapping to multiple Objects. A protocol description of a service is called an RDG (Resource Description Graph). An HTTP GET on the Resource URL of the RDG returns the RDG in W3C-compliant OWL RDF/XML. Because service descriptions are just text documents retrievable by a simple GET, they are readily available for search engine traversal and viewable by browsers[1]. An RDG with input data creates an RIG (Resource Invocation Graph). An HTTP POST of the RIG or a GET with ontology *term=value* assignments in the query string invokes the service. An RIG with output data is called an RRG (Resource Response Graph). Thus SSWAP creates an ecosystem of protocol graphs, all sharing a canonical model, with a common syntax (OWL RDF/XML), under a common services' semantic (SSWAP), amendable to customization by user semantics (adding ontology terms to the Resource, Subject, and Object). SSWAP is a wrapper technology, so it can semantically enable legacy and non-semantic services. Notably, a SSWAP service *description* yields the service amenable to automated semantic *discovery*, *invocation*, and *response*.

Semantic Querying A service's protocol description encapsulates the information needed for its discovery and invocation. Thus one can consider any putative RDG as a query graph (called an RQG: Resource Query Graph) into a knowledge base of all RDGs. For semantic querying, we find all services for which the RQG's: *i*) Resource is a subclass, and *ii*) Subject is a super-class, and *iii*) Object is a subclass, of any service in the knowledge base. Subsumption reasoning covers arbitrary complex, inferred, anonymous classes. The resultant services, and only these services, are guaranteed to be of the type of service queried (or more specialized), to operate on the input data (or generalizations of it), and return data of the requested output type (or specializations of it). This allows us to use a reasoner for match-making based on the output of one service being logically sufficient for the input of another. Thus reasoning is used to examine service descriptions, input data types, and output data types, to enable semantic matching with published services.

Constructing semantic pipelines At `http://sswap.info`, a Web front-end to a backend pipeline manager allows users to connect services into pipelines. Pipelines are built on-demand by using transaction-time reasoning to aid the user in building a workflow of distributed services.

Start with a lexical search Users at `http://sswap.info` may search for services using keywords. Upon selecting a service and adding it to a new

[1] Visit `http://sswap.info`, search for a service, click on 'Service URI' to view the RDG, or visit `http://sswap.info/api/makeRDG` for examples.

pipeline via web-based drag-*n*-drop, we present the user with all downstream services that can operate on the upstream service via semantic querying as described above. In this manner, the user can build a pipeline of services. For each service, we reason over the service's RDG to determine its necessary and sufficient conditions, and based on this construct on-demand a custom user dialog that allows the user to enter the service's required and optional parameters, if any. In a similar manner the Subject is examined, and the user may upload data to be ontologically tagged via the RDG. The protocol declares a datatype property *sswap:inputURI*[2] which allows service providers to write custom Web pages to solicit user input for their services. If *sswap:inputURI* resolves to a Web page, the platform will present that page to the user in addition to allowing the user to use the auto-generated, custom user dialog.

Start with data launched from a web site We provide a Javascript snippet that allows any webmaster to add a "sswap.info" button to their web pages. We call this Web Discovery. We provide a service to allow the Web master to package or reference the data using JSON (see `/api`)[3]. Upon the user pressing the Web Discovery button, the JSON is sent to our Discovery Server, where we translate it into an RDF/XML RQG, perform semantic querying, and present the user with a new pipeline preloaded with their data and the semantic results of all matching candidate downstream services.

Start with the results from previous pipelines Because the last service in a pipeline returns a standard RRG, this can be used to start a new pipeline. In this manner, a pipeline can seed new pipelines. Data is private, but pipelines may be published for public use and are semantically discoverable like services. In this manner, we grow a database of user-built combinations of Web distributed services; this has deep social networking value. We note that public sharing of pipelines does not imply unregulated execution of services: any service is free to gate-keep resources with logins, HTTPS, and so forth.

Pipeline invocation is orchestrated, but execution is distributed RDGs represent published SSWAP services that are offered by third-parties on the Web. When the user initiates a pipeline, we coordinate the invocation and callback of services, but do not ourselves execute the services: the services run independently, asynchronously on their host machines. Downstream services retrieve the upstream RRG from the upstream service with a token and convert it to an RIG without passing through our servers. In this manner we are not privy to non-serialized data being transferred between services, thereby maintaining an important privacy safe-guard.

Transaction-time reasoning SSWAP graphs (RDGs, RIGs, RRGs, and RQGs) are small documents of a few dozen lines of W3C OWL [DL]

[2] *sswap:* prefix resolves to `http://sswapmeet.sswap.info/`

[3] Relative URLs are RESTful endpoints on `http://sswap.info/`

RDF/XML that typically expand to a few thousand triples after first-order reasoning. We use reasoning in four places: *i*) when Providers publish their RDGs with us, we resolve ontology terms by dereferencing them on the Web; we then infer over the closure RDG and store the resulting inferred graph in a triple-store [9]. We use a combination of transaction-time reasoning at publication time and offline processing to maintain the knowledge base; *ii*) when users initiate Web Discovery from a web site by sending us a JSON representation of an RRG, we resolve the RRG, convert it to a RQG, and execute transaction-time semantic querying; *iii*) when users build pipelines we reason during the transaction process to satisfy semantic querying and other pipeline duties; *iv*) when third-party services receive an RIG they need to process the request and return a RRG that complies with the logical contract of their RDG. We provide a kit (`/sdk`) that allows third-parties to run their own servlet reasoner to handle transaction-time reasoning to process requests.

Pipeline management Control is architected as three separate components: *i*) we use Vaadin [10] to offer a RIA (Rich Internet Application) enabling an intuitive, drag-*n*-drop user experience; *ii*) communication to the backend is performed by a 100% RESTful JSON API, making heavy use of idempotent HTTP GETs and PUTs. This means that a user may start building a pipeline, bookmark it, close their browser, and open it anywhere, anytime, and continue their work. It means that users may begin long-running pipelines, and return at their convenience with a different browser and Web session; *iii*) the pipeline manager communicates with the Discovery Server via a RESTful API.

Platform APIs We wrote ~185,000 lines of open-source Java code to build a platform, Java API (`/javadocs`), and helper services. We use the Java API internally, and package it as part of our SDK (Software Development Kit) so anyone may write their own SSWAP services (`/sdk`). Many developers are fluent in JSON, but not in OWL RDF/XML, so we wrote a RESTful translator that allows SSWAP graphs and user ontologies to be written in JSON and then translated to OWL RDF/XML (`/api`; see also `/make` and [11]). We expose Discovery Server engagements as RESTful endpoints (`/wiki/api`).

3 Ontologies

A challenge for the semantic web services is how to enable and incorporate distributed ontologies. We enable the use of user-defined OWL ontologies to allow services to describe their data, and to allow clients to query and engage said services.

Just-In-Time ontologies We used Smart GWT [12] to write an application that allows anyone to host their ontologies on our servers [11]. Users register for a free iPlant account and may create and administer new ontologies (called "namespaces"). Users build ontologies term-by-term using a JSON syntax [13], translate them to RDF/XML with the press of a button, and publish them on-demand. Terms are separately dereferenceable and immediately available to anyone on the web. Just-In-Time ontologies lower the barrier to entry for creating and using small, agile ontologies, but they are not required: ontologies residing anywhere on the web may be freely used, subject to byte and time limits during transaction processing. Ontological statements (*e.g.*, definitions and relation to other terms) are read and used in reasoning if dereferencing term URIs returns OWL RDF/XML statements.

Support for "large" legacy ontologies: module extraction with BioPortal BioPortal [14] is a major repository funded by the National Center for Biomedical Ontology. It contains over 320 ontologies, and over 180 OWL ontologies. We use the method of [15,16] to process each OWL ontology offline to generate "atoms," such that at transaction-time we can compute the subset of statements (called a "module") that are necessary and sufficient for complete entailment over any subset of terms (called a "signature"). Importantly, for moderate sized signatures the module is often much smaller than the ontology itself [15], thus lending it as a key approach to bringing large, legacy ontologies to transaction-time applications in the semantic Web. Currently, ontology modularization is available as a service at `/modularize`. As of this writing, we are implementing a strategy to incorporate it into transaction-time processing but this is not yet part of the larger platform.

Ontologies enable semantic querying and reasoner-assisted semantic pipeline construction When we process a SSWAP graph, we extract ontology terms and dereference them to retrieve their OWL statements. If these documents themselves contain terms, we dereference those, and continue this cascade until closure is achieved, subject to traversal depth, byte, and time limits. For Web Discovery and pipeline construction we then use Semantic Querying (described above) to find matches between data and/or the output semantics of the upstream service and the input semantics of all putative downstream services. Subsumption determination is performed at transaction time, so axiomatic subsumption claims (*e.g.*, `rdfs:subClassOf`) are supported but not required: the reasoner uses transaction-time classification to determine subsumption. Note that it is the SSWAP protocol that makes this possible, because the protocol ensures that the subject and object semantics of RDGs, RIGs, RRGs, and RQGs are comparable.

4 Integrating Enterprise, HPC, and the Semantic Web for Biodiversity

Enterprise resources TreeGenes [7] is a large biological resource serving over 2500 forest geneticists from over 800 organizations. It contains data from 15 yrs on over 1200 species, including genomic, phenotypic, and other data. We wrote 11 SSWAP services to expose slices of this data and added SSWAP Web Discovery to TreeGenes' DiversiTree [17]. For geographically-oriented tree scientists, we wrote a mapping tool called CartograTree [18,19]. Researchers can search specific geographic regions, tree species, phenotypes, or environmental parameters and customize their analysis accordingly. We enabled CartograTree with SSWAP Web Discovery so that scientists can launch directly into semantic discovery. The iPlant Collaborative serves over 7500 scientists with enterprise-class and High Performance Computing (HPC) resources, petabyte-scale storage, and other resources. We wrote semantic pipeline support to engage HPC XSEDE resources [20] and used SSWAP to semantically wrap 10 resources in the domain of multiple sequence alignment and phylogenetic tree reconstruction.

Biodiversity The DiversiTree/CartograTree/SSWAP integration is driven by questions arising from climate change, disease resistance, and conservation. Knowledge of the adaptive genetic potential of forest tree populations is critically important for evaluating their vulnerability to a changing climate [21]. Forests are key to sequestering carbon and consequently contribute an important role to mitigating or reinforcing the impacts of climate change. Healthy forests provide fundamental habitat for valued biodiversity and essential ecosystem services in the form of global carbon cycling, clean water and air, fiber, and recreation. Sustaining healthy forests in the face of climate change is a central challenge for resource management [22]. Towards this goal, researchers are examining candidate loci to understand how individuals and populations are impacted by environmental factors. Specifically, a fusion of population genetics and landscape ecology to layered geographic information systems allows for focused studies of how landscape features affect genetic variation [23-25].

Experimental design often focuses on first identifying candidate genes under selection from geoclimatic factors, determining their allelic diversity, and testing for associations between trees' genotype, phenotype, and the environment. CartograTree connects the TreeGenes' repository of genotype and sequence data to environmental and phenotypic resources. TreeGenes houses approximately 901,000 sequences, 24 million genotypes, and 20,000 phenotypes on individuals from over 1,200 different forest tree species. Sequencing includes either Sanger-based or next-generation approaches, and used to identify polymorphisms in small populations. The polymorphisms are then validated in larger populations through the use of high-throughput genotyping assays. In many cases, genotyped trees are phenotyped for various traits. Barcode identifiers assigned during sample collection are maintained through DNA extraction, sequencing, genotyping, and phenotyping, while also associating trees with their geo-referenced coordinates. The external sources supplying environmental and phenotypic data include relevant portions of the FLUXNET (Ameriflux) [26], WorldClim [27], and TRY-DB [28] repositories. Ameriflux represents 81 remote sensing sites across North and South America; WorldClim is a compilation of five different climate databases covering the globe; TRY-DB enhances phenotypic data with approximately 80,366 geo-referenced phenotypic records represented by 368 species. Within CartograTree, specific queries and filters are available to select by genus, species, or phenotype of interest. The phenotypic selections include economically relevant traits, disease evaluations, and developmental metrics. The map portion of the interface gives users the option to select regions of interest, and capture the associated environmental data, such as slope, elevation, precipitation, seasonal temperatures, and more. From this, scientists can send selected data for SSWAP Web Discovery, for example, to perform multiple sequence alignment and phylogenetic tree reconstruction on high performance computing clusters. A full description of CartograTree and SSWAP is published at [19]. Association studies are facilitated through the ability to create flat files based on the common phenotypic or environmental evaluations for a selection of trees. The results of these studies are aimed at improving land-management decisions through the identification of genotypes that will thrive in specific environments; information that is necessary for reforestation, disease resistance, and climate change.

5 Conclusion

Semantics and biodiversity is still in its nascent years. Our work is focused on a division of scientific labor between domain-specific information resources such as TreeGenes, infrastructural resources such as iPlant, high performance computing assets such as underlying the phylogenetic applications available on XSEDE, and the larger Web. iPlant's Semantic Web Platform is developed as the technological conduit for integration across these resources. It uses transaction-time first-order description logic reasoning to allow semantic web services to be discovered, connected, and invoked via a simple drag-*n*-drop web interface. TreeGenes, DiversiTree, and CartograTree offer an initial foray into the use of these technologies for forest tree biodiversity.

Acknowledgements We thank Pavel Klinov for work on ontology modularization and Yan Kang for work on the Just-In-Time Ontology editor.

This work was supported by NSF grants for the iPlant Collaborative (#DBI-0735191) and SSWAP (#0943879).

6 References

1. `http://www.ncbi.nlm.nih.gov/sites/gquery`
2. Altschul S.F., Gish W., Miller W., Myers E.W. and Lipman D.J. Basic local alignment search tool. J. Mol. Biol. 215: 403-410 (1990).
3. `https://pods.iplantcollaborative.org/wiki/display/DEapps/List+of+Applications`
4. Goecks, J, Nekrutenko, A, Taylor, J and The Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol. Aug 25;11(8):R86 (2010).
5. Goff, S. A. *et al*., The iPlant Collaborative: Cyberinfrastructure for Plant Biology. Frontiers in Plant Science 2, 1 - 16 (2011). DOI: 10.3389/fpls.2011.00034.
6. Gessler, D.D.G., Schiltz, G.S., May, G.D., Avraham, S., Town, C.D., Grant, D., Nelson, R.T.: SSWAP: A Simple Semantic Web Architecture and Protocol for semantic web services. BMC Bioinformatics, 10:309, pp. 1-21 (2009).
7. `http://dendrome.ucdavis.edu`
8. `http://sswapmeet.sswap.info/sswap`; see also `http://sswapmeet.sswap.info/jit/sswap`. Also included at the URL is an additional class and property for asynchronous service invocation.
9. Pellet + Stardog; see `http://stardog.com`
10. `http://vaadin.com`
11. `http://sswapmeet.sswap.info`
12. `http://smartclient.com`
13. `http://sswap.info/api/JSONSyntax`
14. `http://bioportal.bioontology.org`
15. Del Vescovo, C., Gessler, D.D.G., Klinov, P., Parsia, B., Sattler, U., Schneider, T., and Winget, A.: Decomposition and Modular Structure of BioPortal Ontologies, In: ISWC, LNCS, 7031: pp. 130-145 (2011)
16. Klinov, P., Del Vescovo, C., Schneider, T.: Incrementally updateable and persistent decomposition of OWL ontologies. In: Proceedings of OWL: Experiences and Directions Workshop 2012. Klinov, P., Horridge, M. (eds.) Heraklion, Crete, Greece, May 27-28, 2012. CEUR Workshop Proceedings 849 CEUR-WS.org 2012.
17. `http://dendrome.ucdavis.edu/DiversiTree`
18. `http://dendrome.ucdavis.edu/cartogratree`
19. Vasquez-Gross, H.A., Yu, J.J., Figueroa B., Gessler D.D.G., Neale D.B., Wegrzyn J.L.: CartograTree: connecting tree genomes, phenotypes and environment. Molecular Ecology Resources, (2013) doi: 10.1111/1755-0998.12067
20. `https://www.xsede.org`
21. Neale D.B., Kremer A.: Forest tree genomics: growing resources and applications. Nature Reviews Genetics 12, pp. 111–122 (2011)

22. Peterson, D.L.; Halofsky, J.E.; Johnson, M.C.: Managing and adapting to changing fire regimes in a warmer climate. In: McKenzie, D.; Miller, C.; Falk, D., (eds.) The landscape ecology of fire. New York: Springer: Chapter 10, pp. 249–267 (2011)
23. Manel, S., Joost, S., Epperson, B.K. et al.: Perspectives on the use of landscape genetics to detect genetic adaptive variation in the field. Molecular Ecology, 19, pp. 3760–377 (2010)
24. Manel, S., Schwartz, M.K., Luikart, G., Taberlet, P.: Landscape genetics: combining landscape ecology and population genetics. Trends Ecol. Evol. 18, 189-197 (2003)
25. Feder, M., Mitchell-Olds, T.: Evolutionary and ecological functional genomics. Nature Genetics Reviews, 4, pp. 649-655 (2003)
26. Baldocchi, D., Falge, E., Gu, L.H., *et al.*: FLUXNET: a new tool to study the temporal and spatial variability of ecosystem-scale carbon dioxide, water vapor, and energy flux densities. Bulletin of the Amer. Meteor. Soc., 82, pp. 2415–2434 (2001)
27. Hijmans, R.J., Cameron, S.E., Parra, J.L., Jones, P.G., Jarvis, A.: Very high resolution interpolated climate surfaces for global land areas. International Journal of Climatology, 25, pp. 1965–1978 (2005)
28. Kattge, J., Diaz, S., Lavorel, S. et al.: TRY - a global database of plant traits. Global Change Biology, 17, pp. 2905–2935 (2011)