SAF: A Provenance-tracking Framework for Interoperable Semantic Applications

Evan W. Patton[1], Dominic Difranzo[1], & Deborah L. McGuinness[1]

[1] Rensselaer Polytechnic Institute
110 8th Street
Troy, NY, USA, 12180
{pattoe, difrad, dlm}@cs.rpi.edu

Abstract. This paper describes the foundations of a framework for constructing interoperable semantic applications that support recording of provenance information. The framework uses a client-server infrastructure to control the encoding of application. Provenance records for application components, settings, and data sources are stored as part of the final application file using the Open Provenance Model (OPM) [1]. The application can render events such as setting changes to users so that they can identify when collaborators make changes to the application. We demonstrate how the system can be used to collaborate on a project, identify errors in data sources, and extrapolate insights to other data sets by making changes to the application. Lastly, we outline some key issues related to using asymmetric key encryption for tracking changes in semantic content and how we address them (or not) within this framework.

Keywords: provenance, semantic framework, semantic collaboration

1 Introduction

As open linked data continues to proliferate on the World Wide Web, better methods of building applications will need to be established to cover the wide range of issues that rise from maintaining and interacting with this complex network of data. Tracking provenance, both of the underlying data sources and the application processes and configurations, will be critical to maintaining integrity of information and knowledge exchange between individuals, and is a key issue that needs to be solved as part of the growth of the Semantic Web. This paper describes the foundations of an application programming framework that provides a possible solution to maintaining application process and configuration integrity when applications are shared across multiple users.

Since collaboration of data, processes, and users is one of the primary drivers for provenance on the Semantic Web, the Semantic Application Framework (SAF) was designed for building applications that pass data between themselves and track when users make changes to individual settings, resulting in changing application behavior.

Use case. A user should be able to instantiate one or more semantically enabled applications within the application workspace, link those applications together to

form a workflow, and import one or more datasets to explore using the composite application the user generated.

Our approach aims to encode provenance data and the application structure and settings using RDF+XML. , The exported file is signed using asymmetric encryption so that other users can verify the integrity of the application when the receive it. Additional modifications by third-party users must be tracked in order to identify when individuals change the behavior of the application.

2 Related Work

The Semantic Application Framework is aimed to bring workflow-style construction to web-based compositions of data and applications. Rather than thinking of applications as segmented processes completely independent of one another, the SAF treats them as individual components that can be linked together by end users, making data and processes of one application available to another, much like existing scientific workflow systems such as Kepler [2]. Unlike Kepler, SAF-based applications work entirely within the Resource Description Framework, so that any applications that understand a common vocabulary can manipulate and annotate the same data.

One of the goals of this work is to wrap existing web applications with SAF so that users can make use of semantics in existing Web 2.0 tools and record provenance of themselves and others interacting with existing data. Similar advances have been made using plugin models, such as the VisTrails system [3], that provides a socket-based method for applications to record events, replay events, and provide a visual representation of the provenance trace to the user. This has been tested in two environments: radiology [3] and 3D modeling [4]. In both scenarios, a plugin for an existing software tool provides records of user interaction by tracking the undo stack and reporting changes to the VisTrails server. Users can then backtrack by simply choosing a point in the generated revision history tree.

3 Demonstration

We will walk through an example of how the Semantic Application Framework allows different users to collaborate and build dynamic web applications together. In this example Alice and Bob, two students in a 20th century history class, want to build a small web application to visualize the major battles in World War II. Alice starts off wanting to visualize the timeline of battles in the European theatre for World War II, so she creates a SPARQL endpoint application instance, a timeline application instance, and links them together. She then identifies a third party SPARQL endpoint where the World War II information is located for the SPARQL app. She can then visualize the data in the timeline app. She selects a particular time frame, saves the application, and emails it to Bob.

Bob likes the timeline but wants to see more information regarding the progress of the Allies through Europe. He believes a map displaying the location of the battles would help show this. He augments his copy of Alice's original application by creating a second SPARQL application to query for location information and a map application instance to plot the points. He combines the output of the timeline application with his copy of the SPARQL application as input for the map application. By changing the timeline's frame, he can now see how the battle points move through the map over time. He saves the application and sends it back to Alice.

Using the provenance data that SAF collects, Alice can observe the changes in the application Bob made and track what datasets, applications, and users have been added since he emailed it. Alice thinks the modifications Bob made results in a powerful visualization of the European theatre, and wants to use the same system to investigate the Pacific theatre. She updates the SPARQL applications to point to the appropriate endpoints and uses the visualization to identify some key conflicts. She then saves the application and sends it to Bob.

Bob opens the file, and can observe the changes Alice has made to the application. He notices that some of the map locations for the battles don't match up, and that the marker for the Battle of Midway seems to be too far east as compared to his textbook. He decides the data source Alice included for map locations may not be trustworthy, and identifies alternative data source to replace it.

4 Implementation

4.1 Architecture and Security Model

The Semantic Application Framework (SAF) uses a client-server architecture combined with X509 certificates over the secure socket layer (SSL) to enforce provenance tracking. These certificates are used for signing the application so that other users can use the server to verify the authenticity of an application description and the provenance associated with that application.

When an application is exported, the server generates an RDF file that describes the application, the provenance of the application, and a pointer to the server that generated the application. This file is then signed using the server's private key, and the signature is included as a triple. Any other SAF-capable server can reverse this process to verify that a third party has not tampered with the RDF file.

4.2 Client-side Implementation

The client-side portion of SAF is written in JavaScript, and provides a number of high-level classes that provide necessary fundamentals for engineering provenance-tracking applications. Data sources are loaded into the application by way of the RDF parser used by Tabulator [5]. Applications derive from these classes, and use a provided set of function calls to establish mechanisms for both user-application interaction and application-application interaction. Lastly, the implementation is

responsible for providing tools for the user to identify changes to the application, which it does by rendering a trace on the right-hand side of the browser window.

Application-application interaction. For applications to interact with one another, they must establish input and output bindings. A binding is a point where RDF data is supplied to or generated by the application. The user can then link applications together by identifying what bindings should be linked together. When data appears on an output binding, all of the applications with input bindings bound to it are sent a notification for them to respond to the presence of this new data.

User-application interaction. Applications must instantiate an instance of a SAFUserInterface object or derivative that encapsulates an HTML div element where the application can render its interface. The interface renders settings, although the mode of those settings may change from application to application. In the sample application discussed above, for example, the timeline encodes a start date-time and an end date-time through the selection boxes that the user can drag. When a user makes a change to a setting, the application should call the setSettingValue function to register the change for provenance tracking.

5 Summary

In this paper, a framework for the foundation of provenance-supporting semantic applications was discussed with respect to students collaborating on a research project for a class. Such methods, however, can be generalized to any source of semantic data on the World Wide Web. We will present a demonstration using our implementation that shows how applications built within a shared framework can allow users to build collaborative applications and keep attribute and track changes to applications by users.

6 Future Work

There are many elements to this Semantic Application Framework that need to be further developed. The use of asymmetric encryption raises the issue syntactic versus semantic equivalence. The signing mechanism only works if the files are not modified in any way. However, viewing a file then saving it could result in content reordering while maintaining exact semantics. Algorithms must be made more robust to this.

Additionally, SAF does not take advantage of OPM metadata tied to existing data sources. When a user imports a data source, the software does not utilize any existing provenance information stored in that data source. Therefore, it can be difficult to detect when a change to the data source may have occurred that could change the application behavior. Future versions will have to take care to identify OPM information, or follow rdf:seeAlso to look for metadata resources to better capture all

of the available provenance information. We will also investigate using another provenance Interlingua – PML – for more completely encoding inferences.

Lastly, the current implementation works on a single client communicating with a single server. One of the planned extensions to the software involves the composition of applications running on separate servers through the use of WSDL [6] interfaces using OWL-S [7] or similar service markup languages.

References

1. Moreau, L., Clifford, B., Freire, J., Gil, Y., Groth, P., Futrelle, J., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Simmhan, Y., Stephan, E., Van den Bussche, J.: The Open Provenance Model Core Specification (v1.1), http://eprints.ecs.soton.ac.uk/18332/1/opm.pdf
2. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S.: Kepler: an extensible system for design and execution of scientific workflows. In: Scientific and Statistical Database Management, 2004.
3. Freire, J., Silva, C.T., Callahan, S.P., Santos, E., Scheidegger, C.E., Vo, H.T.: Managing Rapidly-Evolving Scientific Workflows. In: International Provenance and Annotation Workshop, 2006.
4. Callahan, S.P., Freire, J., Scheidegger, C.E., Silva, C.T., Vo, H.T.: A Process-Driven Approach to Provenance-Enabling Existing Applications.
5. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D.: Tabulator: Exploring and Analyzing linked data on the Semantic Web. In: Proceedings of the 3rd International Semantic Web User Interaction Workshop..
6. Christensen, E., Cubera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl
7. Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D.L., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K.: Bringing Semantics to Web Services: The OWL-S Approach. In: Semantic Web Services and Web Process Composition. doi:10.1007/b105145