A Publish and Subscribe Collaboration Architecture for Web-Based Information

M. Brian Blake
Department of Computer Science
Georgetown University
Washington, DC, USA
blakeb@cs.georgetown.edu

David H. Fado and Gregory A. Mack
Advanced Systems and Concepts
Science Applications International Corp. (SAIC)
Arlington VA, USA
{David.H.Fado, Gregory.A.Mack}@saic.com

ABSTRACT
Markup languages, representations, schemas, and tools have significantly increased the ability for organizations to share their information. Languages, such as the Extensible Markup Language (XML), provide a vehicle for organizations to represent information in a common, machine-interpretable format. Although these approaches facilitate the collaboration and integration of inter-organizational information, the reality is that the schema representations behind these languages are reasonably difficult to learn, and automated schema integration (without semantics or ontology mappings) is currently an open problem. In this paper, we introduce an architecture and service-oriented infrastructure to facilitate organizational collaboration that combines the *push* features of the publish/subscribe protocol with storage of distributed registry capabilities.

Categories and Subject Descriptors
D.2.11 [**Software**]: Software Engineering: Software Architectures – *domain-specific architectures.*

General Terms
Design, Experimentation, Standardization, Languages

Keywords
Distributed and heterogeneous information management, management of semi-structured data

1. INTRODUCTION
Web data can be defined as a form of data marked-up for Web use, sometimes also called *semi-structured* data. XML is a markup language that provides a universal format for organizations to represent their underlying web data. Objects in XML data are specified using *elements*. A list of elements and their *attributes* represent the fundamental structure for defining an XML schema. A major problem occurs when two organizations develop their schemas independently and produce different XML schemas for essentially the same data or capability. In the future when these organizations need to merge their capabilities, there is a requirement for *schema integration*. Although there are modeling approaches that facilitate integration [3], the idea of schema integration without context information or *semantics* is an open problem. In fact, automated tools for schema integration, when semantics are not present, are practically nonexistent, and human-intervention and organizational collaboration are required.

Instead of addressing the problem of schema integration first, we believe the initial step should be creating a seamless organizational collaboration environment which will facilitate web data sharing. Although registries [2] have typically facilitated data sharing, they lack the capability of real-time distribution of data. We introduce an architecture that extends standard registry capabilities with an integrated messaging infrastructure. Furthermore, the architecture supports a suite of graphical services that allow users to create, store, and activate query instructions on the web data prior to distribution. This architecture can serve as a first enabling step towards a *fully-automated* collaboration architecture, perhaps employing ubiquitous, agent-supported technologies. In the next section, the usage of the architecture is discussed in an applied setting. The subsequent sections discuss the design of this proposed architecture and the underlying implementation.

2. INTELLIGENCE SHARING SCENARIO
Under the sponsorship of the Air Force Research Lab (AFRL), new infrastructures, that support information sharing and advanced analytical collaboration, are being created to support intelligence analysts. Many companies and universities are creating intelligence tools containing analysis information and complex models. These tools publish assets in some form of structured mark-up language. A major challenge in this domain is making these assorted capabilities available on a *single analyst desktop*.

As an example, intelligence analysts typically issue or receive urgent requests for information. These requests spur the formation of a team of perhaps a dozen relevant experts who must construct accurate responses in a short period of time. Due to the nature of this domain, the composition of these communities of interest will evolve as intelligence needs change and the cognitive support tools employed will vary according to the shifting group membership. Such a dynamic environment cannot rely on a strict schema applied to all analysts without undermining the capability to bring in new users, new points of views, and new tools. One key to facilitating collaboration in a dynamic environment is the capability of comparing schemata without merging them, thereby exposing different perspectives among members of the community of interest. The knowledge of varying perspectives accelerates team formation by exposing potential points of miscommunication.

Currently, the team formation and processes are relatively ad-hoc. Using the proposed architecture, an analyst will employ a principled approach using his/her desktop to create and participate in an on-demand, virtual *community of interest*. The proposed architecture supports advanced collaboration by providing data sharing as a first step so that the differences of the schema and the subsequent merging activities can be included as part of the community of interest definition.

3. ARCHITECTURE DESCRIPTION

The publish/subscribe collaboration architecture can be discussed in terms of server-side modules and requester-side modules. A provider organization can register with the system by providing a schema or a specific type of web-based information. A new requester organization can subscribe to the instance data (based on pre-registered schemas) as it is populated on the server. The requester can choose to receive complete data files or a subset of the web-based data. During regular operations, web-based information is distributed in real-time using interface modules on both the requester-side and server-side. On the requester-side, there are two modules that package and unpackage the instance data for transmissions to and from the server. There is a human-driven module, and a module that enables machine-to-machine interface. At operations time, provider organizations will transmit instance data regularly and propagate the subscribed data to requesters while recording the transactions in a local data repository.

4. SHARX PROTOTYPE

A service-oriented, component-based prototype was created that leverages technologies such as Java-based web services, XML querying software, a relational database, and web server packaging techniques. Entitled *Sharx* (i.e. derived from ***SHAR****e* ***X****ML*), the prototype consists of five major high-level components and is packaged as a Java web archive file (i.e. WAR file) easily deployed on a network-accessible server. Once deployed, the initiator uses the *server-side graphical user interface component* developed as a Java servlet to name the community of interest and perhaps enter the initial schemas and instance data. Others invited to the community of interest can also enter schemas and information by hand, or download a pre-configured *client proxy component* to connect their local tools to the *distributed transmission component* (both developed as Java web services) of the server. All communication is captured in communication objects and stored with the *data management component* to a both a relational database and a XML registry. As a service, stakeholders can subscribe for and query against information using the user interface and the *collaboration processing component* that incorporates XQuery techniques. The Sharx prototype is shown in Figure 1.

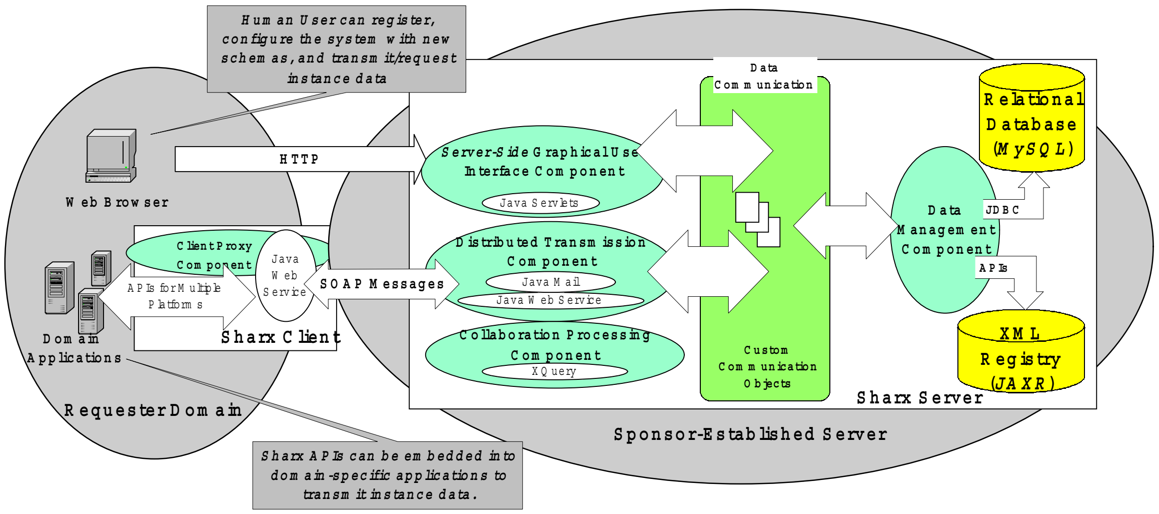
Human User can register, configure the system with new schemas, and transmit/request instance data
Web Browser
HTTP
Server-Side Graphical User Interface Component
Java Servlets
Data Communication
Relational Database (MySQL)
JDBC
Data Management Component
APIs
XML Registry (JAXR)
Client Proxy Component
Java Web Service
APIs for Multiple Platforms
SOAP Messages
Distributed Transmission Component
Java Mail
Java Web Service
Collaboration Processing Component
XQuery
Custom Communication Objects
Sharx Client
Sharx Server
Domain Applications
Requester Domain
Sponsor-Established Server
Sharx APIs can be embedded into domain-specific applications to transmit instance data.

Figure 1. Sharx Architecture Design and Implementation Diagram.

5. DISCUSSION

Probably the most closely related architecture work in the area of assisting human collaboration for schema integration was the blackboard system introduced by Ram et al. [4]. Ram introduces a black-board system for integrating relational schemas. This work is related by nature of the support for human collaboration. The Sharx approach relies on technologies that push information to the stakeholders, whereas the blackboard approach relies mostly on human-driven collaboration. By addressing schemas for web data, Sharx addresses a different domain than that of relational schemas.

6. ACKNOWLEDGMENTS

Students of COSC 346 (Software Engineering II), Georgetown University, Washington DC (Spring 2004) helped in software development. We also acknowledge conversations with Dr. Leo Obrst and Dr. Len Seligman of The MITRE Corporation. This work was partially funded under the AFRL's Topsail program.

7. REFERENCES

[1] Fenkam, P., Kirda, E., Dustdar, S., Gall, H., Reif, G. Evaluation of a publish/subscribe system for collaborative working. In *Proceedings of the WETICE 2002,* June, IEEE Computer Society Press.

[2] Java API for XML Registries (JAXR) (2004): http://java.sun.com/xml/jaxr/index.jsp

[3] Passi, K. et.al. A Model for XML Schema Integration. (In *Proceedings of the EC-Web 2002)* (Aix-en-Provence, France, September 2-6, 2002)

[4] Ram, S. and Ramesh, V. A blackboard-based cooperative system for schema integration. *IEEE Expert*, *10* , 3 (June 1995) 56 – 62