Vocabularies in Collaboration Channels

George Mathew, Zoran Obradovic
Center for Information Science and Technology
Temple University, Philadelphia, PA, USA
{George.Mathew,Zoran.Obradovic}@temple.edu

***Abstract*— Collaborators use vocabulary germane to the domain in context. Collaboration applications and collaborating systems use vocabularies at different (lower) layers that are specific to the state in which they execute. Distributed, yet collaborative domain-specific applications have demonstrated success when lower layer vocabularies are well defined. These standard vocabularies enable platform neutral, programming language neutral and client neutral mechanisms to realize successful handshake between collaboration applications. The concept can be extended to an application neutral, protocol neutral, platform neutral, programming language neutral and client neutral vocabulary model that will facilitate harmonious handshake of collaboration channels. This paper addresses the need for standardizing vocabulary at the collaboration channel level and presents a model for realizing vocabulary-awareness in a generalized neutral format. A pilot study done to implement the model using a sample vocabulary is presented.**

I. INTRODUCTION

A vocabulary is a set of tokens that has specific meaning pertaining to the domain under consideration and helps facilitate effective exchange of information in that domain. Vocabulary is integral to collaboration. An agreed upon vocabulary is imperative to avoid ambiguity and convey right notion in collaborative environments. Participants in a given collaboration use the same vocabulary. Proficiency of the participants in the vocabulary is well pronounced in the case of human entities at the cognitive level. This difference in scale can slide due to various factors (participation, teaching, learning, etc.). At the collaboration application execution level and systems (node) collaboration level, proficiency is based on the subset of vocabulary implemented in the stack. The proficiency can move up in scale if the developers implement a superset of the currently implemented vocabulary stack.

COLLABORATECOM 2010, October 9-12, Chicago, USA
Copyright © 2011 ICST
DOI 10.4108/icst.collaboratecom.2010.61

A vocabulary can be implicit and can function as the base/foundation vocabulary on which to build others. Same schooling leads to implicit vocabulary. For example, principals collaborating in a document management system (irrespective of their technical background, education, etc.) identifies a pdf document by a well known icon. The use of GUI in personal computers provides the common schooling.

A vocabulary can be the defacto standard. <user>@<dns_domain> became the de facto token for email addresses. Addresses with the separators !, % and :: were once in use [1]. The token "<user>@<dns_domain>" is part of the email domain specific vocabulary. The addition of MIME types to email is an example of vocabulary expansion.

A standard vocabulary is a formalized and agreed upon designed vocabulary. Universal domain-specific vocabulary could be rooted in profession/religion/language.

With the proliferation of web services and global e-activities, standardization of vocabularies has been initiated by various institutions. EDI (Electronic Data Interchange) format within the United Nations Economic Commission for Europe (UN/ECE) in the working Party for the Facilitation of International Trade (WP.4) standardized on the vocabulary for international business transactions. The syntax or grammar of this common business language, known today under the acronym UN/EDIFACT, was approved as ISO standard 9735 [2]. In e-science, the VCDE (Vocabularies and Common Data Elements) workspace within caBIG [3] has created standardized vocabularies.

II. VOCABULARIES AT VARIOUS PHASES AND TIERS

The Initiation, Formation and Operation phases of a Collaboration Life Cycle [4] uses different vocabularies. Initiators use a base vocabulary to address the “why tackle this problem?” issue.

Initiators collaborate with facilitators to map an execution plan and team building. The vocabulary used during the collaboration operation phase is domain specific (see Fig. 1). The collaborators use a vocabulary for peer-peer communications. The collaboration applications (tools) that facilitate the collaboration use a different context specific vocabulary. This vocabulary could be completely hidden from the collaborators. The collaboration channels make use of an even lower level vocabulary to exchange information.

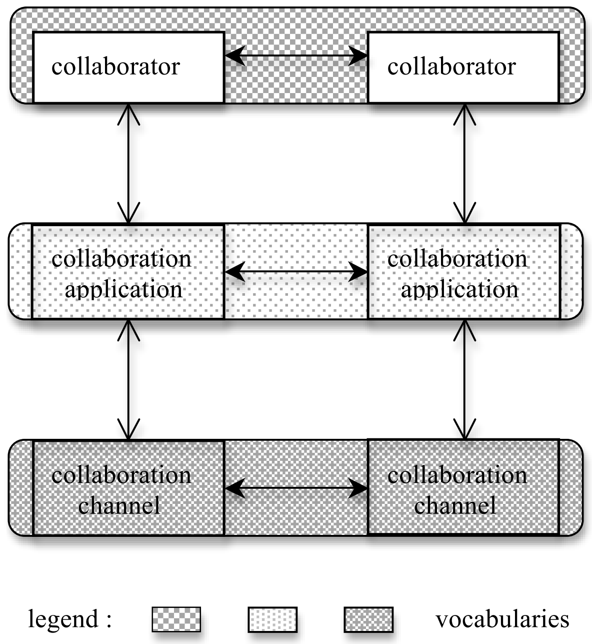

Figure 1. Vocabularies in different tiers

Kock [5] has referred to the vocabulary at the collaborator level as 'mental schemas possessed by the individuals'. UN/EDIFACT is a vocabulary at the collaboration application level.

III. COLLABORATION APPLICATIONS

Principals use CAs (Collaboration Applications) as tools to collaborate with peers. An existing tool might be used for this purpose. Sometimes custom CAs are developed as in the case of TeraGrid [6]. A hybrid approach of retrofitting tools into a collaboration environment is possible using some technologies. Eg: SAML [7]. Zimbra [8] attempts to integrate popular tools (email, IM, calendar, etc.) while Accessgrid [9] attempts to integrate independent tools into a group-group collaboration environment. The CAs themselves are designed and developed based on the requirements and needs of the collaborators. Retrofitting existing applications into a collaborative suite is expensive and not without problems.

Collaboration applications being distributed in nature, requires a well-defined vocabulary at the communication channel level. Traditionally, the protocol level vocabulary has been well defined. SMTP protocol (RFC 821 [10]) and HTTP protocol (RFC 2616 [11]) provides a platform neutral, client neutral and language neutral vocabulary for the protocol. A great deal of efforts has gone into integrating multiple protocols into collaboration suites. A simple chat application integrates text, audio and video. A collaborative groupware supports email, shared calendars and shared resource managers. Even social collaborations (eg. mashups [12] and yahoo pipes [13]) provide glue for integration. These facilities exist at higher levels. Philip's [14] work on semantic similarity is aimed at avoiding ambiguity at natural language level. At the tool level, C-SciPort (see [15]) was designed to help researchers with a centralized thesaurus-aided collaboration application. All these works are at higher levels focused on the principals in the collaboration. In this paper, the focus is on the lower levels of collaboration channels. To our knowledge, there has been no prior work done at the lower levels to integrate vocabularies for collaboration channels. Many protocols that once existed as the basis for standalone applications have evolved to become part of collaboration suites. A collaboration portal can have email, instant messaging, calendar, white pages, soft phone, etc. integrated within a single framework. Because of this trend towards integrating more and more collaboration channels, a common vocabulary at the lower levels will aid in smoothing out the bumps in integration work.

Given the popular use of different operating systems (Windows, MacOS, linux, etc.), multiple languages (JAVA, C#, C++, Python, PHP etc.), various protocols (HTTP, IRC, SMTP, FTP, etc.) and client accesses (web or desktop); coupled with the distributed nature of collaboration applications, a framework for vocabulary at the collaboration channel level is significant. Middleware technologies (eg. DCE, CORBA, Web Services, etc.) add another layer of integration factor. Integrating heterogeneous technologies with no common grounds require costly development effort. Achieving transparency is even more challenging. As was noted in the 1st System of Systems symposium, "The diversity of terminology is a barrier to conversation. We need to develop a working lexicon to facilitate better understanding" [16].

IV. STANDARD VOCABULARY FOR COLLABORATION CHANNELS

Historically, UNIX systems provided a collection of error codes and messages in a header file 'errno.h'. This provided a protocol-neutral de facto standard for specifying error codes and messages. Protocol-specific status codes and message codes were defined in RFCs. Eg. RFC 977 for network news transfer protocol, RFC 821 for SMTP. These define vocabulary at the protocol level. JAVA exceptions [17] provide a language-specific vocabulary. The JNDI facility in JAVA allows a level of abstraction to interact with various directory services. But, this is limited to directory services and not available in other languages.

Since collaboration channel edges could be housed in any platform, environment or language, a neutral vocabulary that can be used across any parameter of interest will alleviate communication barriers. A generalized framework to achieve this should use a format neutral to the parameters for structuring the information. Various generalized categories can be used to provide the necessary groupings for functionalities. For example, consider 'resource exchange' between collaboration channels. The following table illustrates the protocol-specific versions of resource exchange:

Table 1. Similar features in various protocols

Protocol	Resource	Error #	Message
FTP	file	550	no such file
HTTP	html page	440	not found
LDAP	entry	32	no such object

A parameters-neutral generalization of this could simply be: <message #, 'No such resource'>. Similar generalizations for 'channel initialization' tasks can be done. An email MTA (Mail Transport Agent) server may be trying to connect to another MTA that is unavailable. Or, the user credentials presented for establishing a channel may be invalid. Nosek [18] has characterized this generalization from a tools-neutral perspective as Collaboration Envelopes™ Level 1. A Level 1 Collaboration Envelope™ supports data sharing, but in a way that is non-tool-centric and more of a natural wrapper around sharing. In this paper, data sharing is implied for vocabulary data.

V. A MODEL FOR IMPLEMENTATION

A structural specification for a parameter neutral implementation can be given as follows:

```
<channel_vocabulary> ::= <category> {<category>}
<category> ::= <message> {<message>}
<message> ::= <id> <description>
<id> ::= <integer>
<description> ::= <text>
```

A parameter-neutral implementation can be done in different ways. A sample prototype implementation (using 3 categories – channel initialization, resource exchange and normal operations) using XML is outlined below:

```
<channel_vocabulary>
  <channel_initiation>
    <message>
      <id>0001</id>
      <description>OK</description>
    </message>
    <message>
      <id>0002</id>
      <description>TIMED OUT</description>
    </message>
  </channel_initiation>
  <resource_exchange>
    <message>
      <id>1001</id>
      <description>OK</description>
    </message>
    <message>
      <id>1013</id>
      <description>
        NO SUCH RESOURCE
      </description>
    </message>
  </resource_exchange>
  <normal>
    <message>
      <id>2001</id>
      <description>OK</description>
    </message>
  </normal>
</channel_vocabulary>
```

It is obvious that other categories can easily be added to the xml description (suggesting an extensible implementation of the structural specification of this vocabulary using xml). Any implementation of the structural specification should allow for the expansion of vocabulary definitions to add new categories.

Fig 2. shows how the collaboration channels will interact using this model in a working environment. The library routines should parse the XML information.

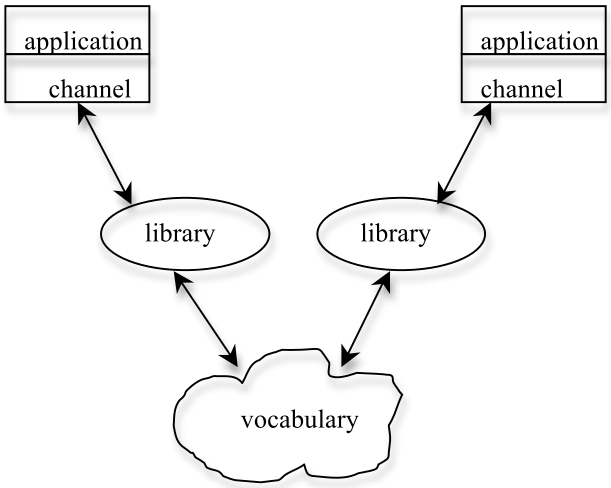

Figure 2. Vocabulary Utilization

The vocabulary collection can be implemented as a centralized model or a distributed model with delegation of authority for domain-specific elements. Note that the distributed model will require an integration point for adding new domains into the main trunk.

A prototype snippet of C code using the library is as follows (interpret_vocabulary() is a library call):

```
main()
{
     CONNECTION_PROFILE *service;
     CHANNEL *channel;
     RESOURCE *resource;
     int ret;

      channel  = channel_open(service);
      ret = getresource(channel,resource);
/*--- assumed that 0 is returned on success ---*/
      if ( ret )
          interpret_vocabulary(ret);
}
```

Figure 3. Snippet of code in C showing incorporation of library call to interpret the vocabulary

A corresponding prototype snippet of JAVA code using the library is similar to the following:

```
public static void main(String args)
{
     ConnectionProfile service
          = new ConnectionProfile();
     Channel channel;
     Resource *resource;
     int ret;

      channel  = channelOpen(service);
      ret = channel.getResource(resource);
/*--- assumed that 0 is returned on success ---*/
      if ( ret )
          interpretVocabulary(ret);
}
```

Figure 4. Snippet of code in JAVA showing incorporation of API call to interpret the vocabulary

VI. FEASIBILITY STUDY

An implementation was done to study the feasibility of these concepts. The study was conducted in two phases. The first phase was to verify that this is an implementable mechanism. In order to ascertain the implementablity, a gateway model was made use of. The gateway model was successful and so a native implementation was done. The native implementation also proved to be successful. An outline of the feasibility study is furnished below.

A JAVA program was written to connect to a mySQL [19] database. The program had tests for the following conditions:

server timeout
invalid user credentials
no such resource
invalid resource format
ok

An xml file was created with the <id,description> pairs matching these conditions. In the initial viability study, the xml file had values for id's in sequential order. The purpose of this phase was just to test the hypothesis without modifying source code for the backend mySQL database. A JAVA method was written to do a cross-match between the value returned by the mySQL database and the value in the xml file. (This is the gateway method between the mySQL database and the vocabulary data.) The library API was written to replace the method that makes the connection to the database (viz. the getConnection() JDBC call). The new method made use of the gateway method for cross-matching. For example, mySQL error code 1049 (suggesting INVALID SERVER NAME) was cross-matched with the code (id 003) in the xml file. The main program needed only one feature change. This change was essentially to call the getConnection() method from the new library API. The program worked as expected.

Based on the success of the gateway API, the next natural progression was to try the API natively. In order for the message id's returned by mySQL to match the ones in the xml file, the source code for mySQL had to be modified. The source code for mySQL (version 5.1.40) was used for this purpose. It was found that all message identifiers that had to be modified were recorded in the single file ‘errmsg.txt’. The message identifiers in this file were changed to match with the message identifiers in the xml file. The source code was compiled and deployed as the mySQL server for this stage of the experiment. The library API was rewritten to do a native xml parsing of the vocabulary. The main JAVA program was recompiled and executed. The results were consistent and as expected.

VII. CONCLUSION

Collaboration channels being distributed in nature, the environment of the end points can neither be pre-determined nor be controlled. Consequently, it is necessary to have an open shared vocabulary. Given the heterogeneous environments the collaboration channels have to participate in, it is important to have a consistent vocabulary to avoid ambiguities. A model for implementing a common vocabulary for collaboration channels was presented. The model is application neutral, protocol neutral, platform neutral, programming language neutral and client neutral. Extending the vocabulary to more categories can be accomplished. Actual experiment was conducted to study the feasibility of an implementation. The experiment done suggests that the formulations proposed in the paper can be implemented in real applications. This is encouraging and provides an incentive for looking into expanding the study to integrate more collaboration channels.

REFERENCES

[1] D. Frey and R. Adams, *!%@:: A Directory of Electronic Mail Addressing & Networks*, O'Reilly & Associates Inc., Sebastopol, CA, USA, June 1994, pp. 8-13.

[2] ISO 9735, International Standard for UN/EDIFACT, http://www.unece.org/trade/untdid/download/r1244r1.doc, June 1997

[3] caBIG, Cancer Biomedical Informatics Grid, http://cabig.nci.nih.gov

[4] H. Tellioğlu, “Collaboration Life Cycle”, in *Proceedings of the 2008 International Symposium on Collaboration Technologies and Systems,* Irvine, CA, USA. pp. 357-366.

[5] N. Kock, *Emerging E-collaboration concepts and applications*, Cybertech Publishing, Hershey, PA, USA, 2007. pp 6.

[6] TeraGrid, http://www.teragrid.org

[7] Online Community for SAML OASIS standard, http://saml.xml.org

[8] Zimbra for messaging and collaboration, http://www.zimbra.com

[9] Access Grid, http://www.accesgrid.org

[10] Simple Mail Transport Protocol (SMTP), http://ietf.org/rfc/rfc0821.txt, 1982

[11] Hypertext Transport Protocol (HTTP), http://ietf.org/rfc/rfc2616.txt, 1996

[12] Mashup (web application hybrid), http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)

[13] Yahoo Pipes, http://pipes.yahoo.com

[14] R. Philip, “Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language”, *Journal of Artificial Intelligence Research*, Vol 11, pp. 95-130, 1999

[15] F. Wang, C. Rabsch, P. Kling, P. Liu, and J. Pearson, “Web-based Collaborative Information Integration for Scientific Research”, *IEEE 23rd International Conference on Data Engineering*, 15(20), pp. 1232-1241. 2007

[16] S.W. Popper, S.C. Bankes, R. Callaway and D. DeLaurentis, “1st System of Systems Symposium: Report on a summer conversation”, November 2004, pp 8. http://www.potomacinstitute.org/academiccen/SoS%20Summer%20Conversation%20report.pdf

[17] Exceptions in Java, http://java.sun.com/docs/books/tutorial/essential/exceptions/definition.html

[18] J.T. Nosek, “Collaborative Sensemaking Support: Progressing from Portals and Tools to Collaboration Envelopes™”, *International Journal of e-Collaboration*, 1(2), April-June 2005. pp 30.

[19] mySQL opensource database, http://mysql.org