Annotating UDDI Registries to Support the Management of Composite Services

M. Brian Blake, Michael F. Nowlan, Ajay Bansal, and Srividya Kona

Department of Computer Science
Georgetown University
Washington, DC
202 687-3084

{mb7,mfn3,ab683,sk558}@georgetown.edu

ABSTRACT

The future of service-centric environments suggests that organizations will dynamically discover and utilize web services for new business processes particularly those that span multiple organizations. However, as service-oriented architectures mature, it may be impractical for organizations to discover services and orchestrate new business processes on a daily, case-by-case basis. It is more likely that organizations will naturally aggregate themselves into groups of collaborating partners that routinely share services. In such cases, there is a requirement to maintain an organizational memory with regards to the capabilities offered by other enterprises and how they fit within relevant business processes. As a result, registries must maintain information about past business processes (i.e. relevant web services and their performance, availability, and reliability). This paper discusses and evaluates several hybrid approaches for incorporating business process information into standards-based service registries.

Categories and Subject Descriptors

D.3.3 [**Programming Languages**]: Language Contructs and Features – *abstract data types, polymorphism, control structures.*

General Terms

Performance, Design, Experimentation, Security, and Standardization

Keywords

Keywords are your own designated keywords.

1. INTRODUCTION

Service-oriented computing [10] promotes the development of modular domain-specific capabilities that can be advertised to and shared among collaborating organizations. Moreover, the syntactic and semantic metadata that accompanies these services [9][11] enable the discovery of these capabilities, on-demand. Discovery, in this environment, largely depends on the accessibility and capabilities of the repositories for which these services are stored. Universal Description, Discovery, and Integration (UDDI) is the leading specification for the development of service-based repositories or registries. UDDI registries of the future should facilitate fast search and discovery of relevant web services akin to the performance currently associated with resolving a domain name. Although, performance and federation are two important aspects of UDDI, an additional requirement for UDDI should be effective process-oriented storage and retrieval. Currently, the abilities to browse and discover independent services as characterized by their overarching business name and/or their capability name are important fundamental operations. However, as service-oriented architectures mature, composite services (i.e. capabilities based on the workflow composition of multiple atomic services) will also be important to persist and manage. UDDI currently has limited support for managing business processes [6]. Although not all federated service registries will need business process annotations, we suggest that a subset of registries frequently used by partnering organizations would benefit from maintaining historical process information.

Currently, there are numerous languages and protocols that support the specification and execution [3][4][5][9]of composite web services. Unfortunately, techniques for incorporating the underlying process information into UDDI registries are limited. In this work, we address several questions as listed below.

- *What are the relevant descriptive attributes of composite web services that must be represented in web service registries?*
- *What are the relevant use cases for process-oriented service registries?*
- *What are the state-of-the-art methods for incorporating process information into registries and the corresponding challenges?*
- *What are the most efficient and effective approaches, both qualitatively and quantitatively, for process management of services in UDDI registries?*

This paper proceeds in the next section with a survey of related work and a discussion of the state of the practice. The next section formalizes a composite service by detailing the most relevant descriptive attributes. Next, we describe the requirements of a process-oriented registry. We next introduce

SAC'09, March 8-12, 2009, Honolulu, Hawaii, U.S.A.

several hybrid approaches for adding process information to web service repositories. Finally, we describe a case study and experimental evaluation of both approaches.

2. RELATED WORK

Universal Description, Discovery and Integration (UDDI) describes the data model associated with a web-accessible registry for the storage and management of web services. Three core hierarchical objects specify the service provider (businessEntity), the web service (businessService), and information about how the service is binded (bindingTemplate). These foundational objects can be extended with the use of technical models or tModels. TModels facilitate the further description of businessEntities, businessServices, and bindingTemplates through classification based on metadata. Each tModel of a businessService represents a certain behavior or classification system that the service must implement. An example would be a web service that takes a state abbreviation as input. This web service would probably choose to reference the tModel that represents the US-State abbreviation classification system. A person looking at the service's bindingTemplate would then be able to see a reference to the US-State System and know that a state should be entered with its abbreviation. UDDI also supports other structures called keyedReferences that allow previously mentioned core objects to be associated to tModels. KeyedReferences consist of a tModelKey, keyValue, and keyName. The tModelKey identifies the referenced tModel. The keyValue allows a categorization of the link between the core object, and the tModel and the keyName is a text string readable for humans. In UDDI v3, keyedReferences can be aggregated with keyedReferenceGroups.

The strength of the tModels and keyedReferences is that further information about the main UDDI objects (i.e. businessEntities, businessServices, and bindingTemplates) are not populated in the repository. Tmodels merely point to web-accessible documents. This paradigm both reduces maintenance of the registry and promotes overall robustness. However, this paradigm also makes it difficult to associate web services that are stored in the registry, which is a necessary requirement for describing business processes within the registry.

The most common research projects tend to address the problem of federating UDDI registries [1] [2][12], although, of most relevance to our work, are the projects that directly address the problem of business process annotations that associate services. Perhaps the leading approach to inserting business processes is the construction of a tModel classification system that mirrors a particular taxonomy of business processes. These tModels can then be used as pointers to the corresponding business process description documents. In industry, several OASIS technical reports [15] [17] describe high-level approaches to integrating tModel classifications with ebXML and BPEL4WS descriptions. Other research projects detail specialized domain-specific methods that leverage the same basic approach [6] [13] These are reasonable approaches, but all business descriptions are defined by external business process documents that are decoupled from the individual services. In fact, since services may be captured at individual locations, replacing services or even discontinuing the offering of a particular business process becomes difficult. In these cases, centralizing some vital part of the process information can be valuable. Luo et al. [8] and Srinivasan et al. [14] use semantic notations embedded in UDDI tModels to associate services. This approach allows for more rich definitions, but the underlying limitations caused by distribution also apply. Other approaches look at the development of external, integrated software mechanisms that run parallel with the UDDI registry [18] [19]. These approaches tend to depart from the essence of the SOA paradigm as they promote proprietary, less standardized solutions.

In this paper, we experiment with a combination of external process documentation and annotations that are embedded directly in the UDDI registry. Prerequisite to any solution, it is important to understand the required aspects of the web services-based business process

3. ANATOMY OF WEB SERVICES-BASED PROCESSES

Web-services based business processes also referred to as *web service workflow*s are similar to traditional processes that are established between human stakeholders. Figure 1 illustrates the metamodel of a web services-based business process using a Unified Modeling Language (UML) class diagram. A difference is, as opposed to human-managed tasks or *steps*, web services enact the underlying steps. A web services-based business process, B_P, contains *user data endpoints*, D_E, defined below.

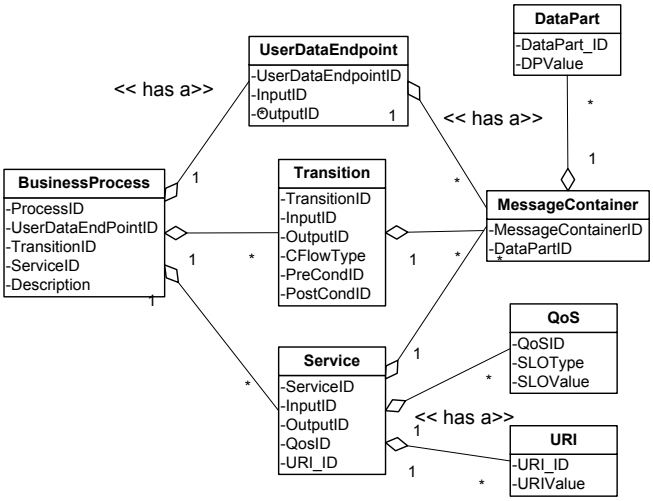

Figure 1. Metamodel of a Web Service-Based Business Process.

Definition (UserData EndPoint):

The user data end point is defined as a pair $D_E = (I_D, O_D)$ where I_D represents the *input information* of the business process provided by the user and O_D represents the *output information,* ultimately generated by the completion of the business process. The business process also has a sequence of tasks (realization of the steps) that are implemented by a set of *services*, $\Omega = \{S_1, S_2, S_3, \ldots S_n\}$. Each service S_i has its own input, I_{Si}, and output, O_{Si}, information; however the set of all input/output information of a service is less relevant than the subset of inputs and outputs that are relevant to the business process. In addition, a service can also be defined with its quality of service information and its URI location.

Definition (Service):

A service is a tuple of its inputs, outputs, QoS parameters, and its URI location. $S = (I_S, O_S, Q_S, U_S)$ is the representation of a

service where I_S is the *input* list, O_S is the *output* list, Q_S is the list of *quality of service* parameters and U_S represents the *URI location*. Each step in the business process is defined by a transition, *T*, that defines the shared information between the output, O_T, of the preceding step that connects to the input, I_T, of the subsequent step.

Definition (Transition):
A transition is represented as a tuple of its inputs, outputs, flow-type, pre-conditions, and post-conditions. $T = (I_T, O_T, F_T, C_{Pre}, C_{Post})$ is the representation of the transition where I_T is the *input* list, O_T is the *output* list, F_T is the flow type, C_{Pre} represents the *pre-conditions* of the transition and C_{Post} represent the *post-conditions* of the transition. Ultimately, the business process can be formally defined as follows:

Definition (BusinessProcess):
A *BusinessProcess* is defined as a tuple $B_P = (D_E, \Omega, \Gamma)$ where D_E represents the user data endpoint, Ω is the set of services involved in the business process, and Γ is the set of transitions in the workflow .

$D_E = (I_D, O_D)$;
$\Omega = \{S_1, S_2, ..., S_n\}$ where $S_i = (I_{Si}, O_{Si}, Q_{Si}, U_{Si})$, for all $i = 1$ *to* n;
$\Gamma = \{T_1, T_2, ..., T_n\}$ where $T_i = (I_{Ti}, O_{Ti}, F_{Ti}, C_{Prei}, C_{Posti})$, for all $i = 1$ *to* n.

The following conditions should hold for a valid business process:

1. For all $S_i \in \Omega$ and $T_i \in \Gamma$, the inputs of S_i are subsumed by the inputs of T_i, i.e., $I_{Si} \subseteq I_{Ti}$
2. For all $S_i \in \Omega$ and $T_i \in \Gamma$, the outputs of S_i subsumes the outputs of T_i, i.e., $O_{Ti} \subseteq O_{Si}$
3. For all T_i, $T_{i+1} \in \Gamma$, the post-conditions of T_i imply the pre-conditions of T_{i+1}, i.e., $C_{Posti} \Rightarrow C_{Pre\ i+1}$
4. For all T_i, $T_{i+1} \in \Gamma$, the outputs of T_i along with the user data inputs of the business process subsume the inputs of T_{i+1}, i.e., $(O_{Ti} \cup I_D) \subseteq I_{T\ i+1}$
5. The inputs of the business process subsume the inputs of the first transition, T_1 (where $T_1 \in \Gamma$), i.e., $I_{T1} \subseteq I_D$
6. The outputs of the business process are subsumed by the outputs of the last transition T_n (where $T_n \in \Gamma$), i.e., $O_D \subseteq O_{Tn}$

The cardinality of data endpoints, services, and transitions vary for each step, such that is necessary to develop *containers* to aggregate the information into sets. The notion of containers is central to business process languages, such as BPEL4WS and BPML [3][4], for aggregating information related to subprocesses.

4. BUSINESS PROCESS AND SERVICE REGISTRIES

In order for organizations to understand their business processes defined with web services, it is important that their process databases include relevant process information as defined in the previous section. Organizations should be able to generally access process information in addition to the service-specific details.

4.1 Potential Use Cases

There are several functions required by a registry that supports composite services as business processes. Figure 2 illustrates the functions of such a repository depicted as a UML use case diagram. The basic registry actors follow the SOA paradigm (i.e. service providers and consumers). Incorporating business process metadata into the registry also supports the interaction of intelligent software components or *agents* to autonomously maintain the integrity of the information. Service providers should be able to insert one or more services into the registry. Service consumers should be able to either browse or explicitly search the repository based on several attributes such as the name/type of business, service, or process. Although only available using specialized approaches, consumers may also want to search by service/process message names. We focus on three major features of such a repository.

- *Advertising a set of services aggregated as a process*

When partnering organizations decide to share services, there may be a predefined understanding for orchestration. As such, these organizations must insert their relevant services into shared UDDI registries annotated by process-based information (i.e. the underlying control flow and data flow).

- *Discovering services associated with processes and discovering processes associated with services*

Current UDDI registries facilitate browsing of services by business name and by service name. A process-oriented registry will also support browsing by process name or type. Consumers should also be able to find all services associated with a particular process.

- *Managing process information by software agents*

Once process information is annotated into a registry, intelligent agents can regularly check the health of the underlying services. Agents can look for indexing configurations that best support the storage of the process information. In addition, agents can record QoS information supplied by service consumers. This QoS information can represent individual services or the process as a whole.

4.2 Alternative Hybrid Approaches

By extending and leveraging the UDDI specification, we have identified several approaches for annotating business processes within service registries. Every service in a UDDI registry has a bindingTemplate structure, which stores references to tModels. TModels are "sources for determining compatibility of Web services and keyed namespace references" **Error! Reference source not found.**. This means that tModels identify how to interact with a web service by describing the technologies it implements. Although, the UDDI registry usually implements default tModels, such as the State Abbreviation System, it is also possible for an administrator of the registry to create tModels, on-demand. There are two approaches for using tModels to describe web services-based business processes.

• *Annotating business process information directly into the UDDI registry*

A new tModel is created for every business process that is identified in the registry. In addition, a *parent* tModel is created that simply classifies any tModel that annotates a business process as such. In this way, when a new process chain is added or identified in the registry, a tModel that points to the parent tModel is created to represent the process. Furthermore, the categoryBag element is used to store references to all the processes of which the service is a part. Figure 3 shows an example tModel. Notice that the keyName of the keyedReference contains the service name and additional control flow information. Also, the keyValue maintains the sequence number of the service in the process. Figure 4 demonstrates the actions taken by the registry to identify and store the existence of a composite web service.

- *Defining business process information using external markup documents*

Figure 5 details the steps for annotating a business process within the registry using the UDDI data structures. Perhaps the leading approach in related work is the use of an external document (e.g. BPEL4WS and ebXML) to store the process information. This method involves simply adding an entry to each of the process documents associated with the relevant services. The document could exist centrally on a main server, or locally with each service provider.

```
<tModel tModelKey="176a3131-0c20-45d1-b31d-efb4f61b8b14">
  <description>This tModel represents the process starting with
   firstService and ending with thirdService. </description>
  <categoryBag>
    <keyedReference keyName="uddi:Process-Representing"
     keyValue="categorization"
     tModelKey="uddi:uddi.org:categorization:types"/>
  </categoryBag>
</tModel>

<businessService>
   <name>firstService</name>
   <categoryBag>
    <keyedReference
      tModelKey="176a3131-0c20-45d1-b31d-efb4f61b8b14"
      keyName="SERV=firstService, TRAN = Sequence"
      keyValue="1"/>
   </categoryBag>
 </businessService>

<businessService>
   <name>secondService</name>
   <categoryBag>
    <keyedReference
      tModelKey="176a3131-0c20-45d1-b31d-efb4f61b8b14"
      keyName="SERV = secondService, TRAN = Sequence"
      keyValue="2"/>
   </categoryBag>
 </businessService>

<businessService>
   <name>thirdService</name>
   <categoryBag>
    <keyedReference
      tModelKey="176a3131-0c20-45d1-b31d-efb4f61b8b14"
      keyName="SERV=thirdService, TRAN = Sequence"
      keyValue="3"/>
   </categoryBag>
 </businessService>
```

Figure 3. Sample Process-Oriented TModel.

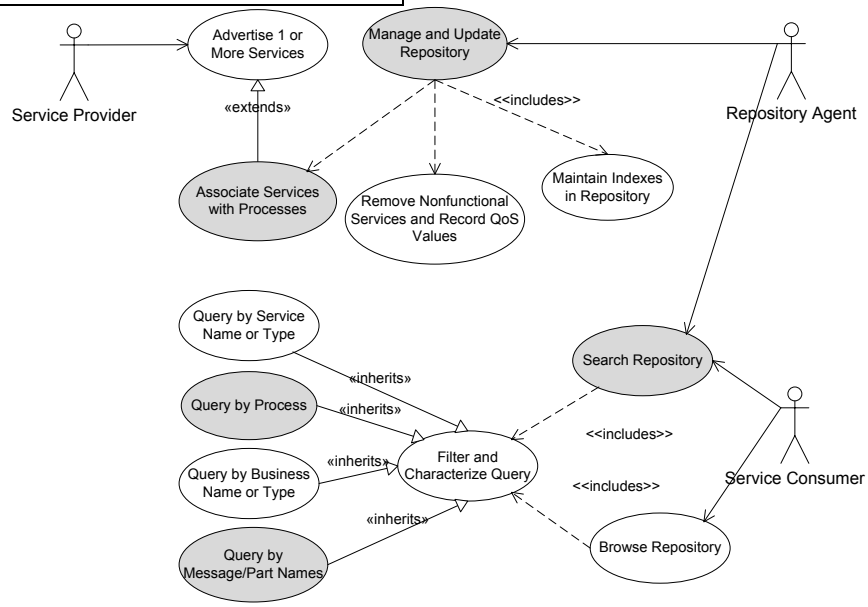

Figure 2. Process-Oriented Service Repository Use Cases
*(** Shaded areas are not currently well supported in the state-of-the-art)*

Annotate(V_S):	Annotating Function
V_S	Vector of serviceKeys for services in chain
G_{SFIRST}	The first service in the vector
G_{SLAST}	The last service in the vector
TM	The newly created process-tModel

TM_{KEY}	The newly created tModel's key
PM	The tModel Process-Classification System
CB	The CategoryBag of an individual structure
S	An individual service
KR	KeyedReference – A pointer to a tModel

$A(V_S, TM)$	Function to add a KR to CB of Services
$MakeTM(G_{SFIRST}, G_{SLAST})$	Function to make a new TM

```
A(V_S, TM)
    KR = TM_KEY
    forAll S in V_S
        S.CB += KR
        return V_S
MakeTM(G_SFIRST, G_SLAST)
        TM.CB += PM
        TM.description = G_SFIRST, G_SLAST
        return TM

Annotate(V_S)
    TM = MakeTM(G_SFIRST, G_SLAST)
    return A(V_S, TM)
```

Figure 4. Annotating a Business Process of Web Services within UDDI Data Model.

$Annotate(V_S)$:	Annotating Function
$MakeEntry(V_S)$	Function to make a new TM
$A(V_S, E)$	Function to add entry to XML Doc.
V_S	Vector of serviceKeys for services in chain
E	An entry containing a list of services
S	An individual service
$S.KEY$	Service key
D	A service's description element
F_S	A service's XML document file

```
A(V_S, E)
    forAll S in V_S
        S. F_S += E
    return V_S

MakeEntry(V_S)
    forAll S in V_S
        E += S.KEY
    return E

Annotate(V_S)
    forAll S in V_S
        S.D = F_S
    E = MakeEntry(V_S)
    return A(V_S, E)
```

Figure 5. Leveraging External Process Documents

4.3 Alternative Hybrid Approaches

Both of the hybrid approaches also have other functions relevant to a business-oriented registry.

• *Service Deletions or Changes when using UDDI Structure Method*

As the registry is updated when a process is identified, it will also change when a service is removed or replaced. These actions cause the process workflow chain to be incomplete. In this case, all the services in the chain are affected and they must modify their process annotation method (either XML document or CategoryBag) by removing the process that is broken. When a service is deleted, all the tModels that it points that *represent composite web services* (excluding simple classifications represented in the bindingTemplate) must be deleted. Any other services in the registry that reference these tModels must remove the reference from their categoryBags. In the future, registries may be able to search for replacements services as opposed to deleting the process.

• *Service Deletions or Changes when Using External Process Documentation*

External process documents contain one entry for each process. Irrespective of the process document notations, each entry will define the serviceKeys of the services in the particular process. In this approach, these serviceKeys are used to find all services that share a process with the deleted service. The registry then edits the relevant XML documents by removing the entry that corresponds to the broken process.

• *Query for Processes by Service Name using UDDI Data Structure Method*

The last action that can be taken on a registry, and perhaps the most desired ability, is the ability to aggregate the information and list of services for every process in which a particular service plays a role. The serviceKey is used to get the service's categoryBag. The categoryBag contains references to all the tModels, that represent all relevant processes. The registry is then queried to find all services which point to any one of the tModels in their categoryBags. The chain of services is then sorted by the tModels and is returned as process.

• *Query for Processes by Service Name using External Process Documentation*

The external document method is quicker because it does not require querying the registry. The XML document for a service is retrieved. In this document is a list of entries, each pertaining to one composite web service. Each entry contains the serviceKey for each service in that particular chain. These services can be retrieved with a single call to the registry by compiling a vector of the serviceKeys. The retrieved service information is then sorted by the order of the entries in the initial service's process document and displayed.

5. CASE STUDY: A GENERAL UDDI BUSINESS EXPLORER

Once process information is associated with or incorporated into a UDDI registry, new graphical user interfaces can be created to support enhanced service discovery and manipulation. As a part of this work, we designed and experimented with a new user interface front-end (i.e. UDDI-P) to the jUDDI registry **Error! Reference source not found.**. A screenshot of the design of the interface is shown in Figure 6. Using this new interface, a user has the capability of browsing known process names as shown on the left side of Figure 6. Once a process is identified, the consumer can further decide to analyze the process to see if it meets the organization need. The interface can display parameters such as estimated delivery date and price range based on service level objectives associated with the process stored in the registry. By embedding process information into the registry, classification of processes can occur to the point where they themselves function as individual services. On the right side of the interface, the user has the ability to browse each process meeting the search criteria by price and delivery. This sort of interface would enable a separate interface that allows the services to be executed.

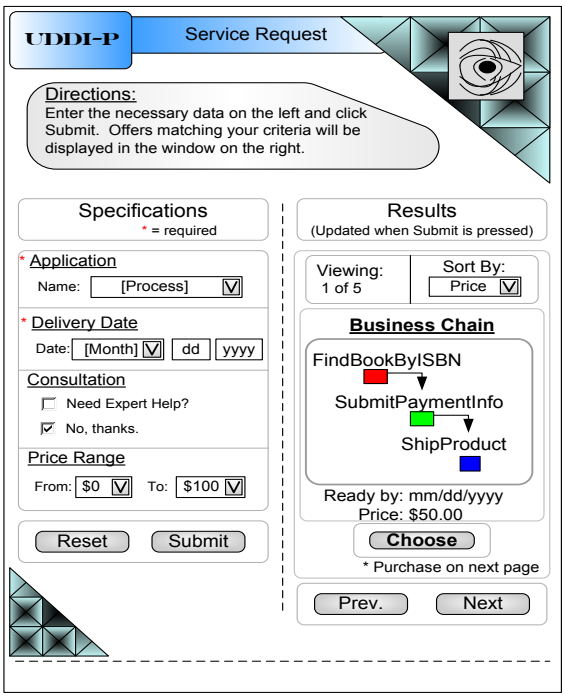

Figure 6. UDDI-P: A Prototype User Interface for Process-Based Service Management.

6. EXPERIMENTATION & EVALUATION

In previous sections, two distinct hybrid approaches for aggregating UDDI services into business processes were introduced. The leading approaches in published works suggest using external mark-up documents to describe business processes. An alternative approach is incorporating specific business process information directly into the registry. We experimented to characterize the performance of these approaches on our prototype repository. The performance is illustrated in Table 1 and Figure 7 based on the use cases illustrated in Figure 2.

In general, the difference in performance between adding individual services, adding a chain, and annotating services with business process annotations were only negligibly different between the two approaches. However, deleting a service was approximately three times faster when external business process documents were used. Another variation in performance is associated with retrieving all service IDs associated with a particular process (or aggregating the services). The aggregation time increased linearly with the increase in size of the UDDI registry embedded process information. The main reason for disparity between the approaches is due to the fact that the latter approach requires a significant query within the repository. This approach must retrieve the categoryBag for all services in the repository and search those structures for a particular keyedReference. The external business process document does not require this step since the process information for a service is stored in one location: the XML-based document. The retrieval of this document is much faster when compared to the query that must take place within the registry. Although the performance for an external file is more favorable, having external BPEL4WS files causes the duplication of process information (i.e. the same process entry could appear in multiple files that may be attached to the underlying services). Depending on the management of the BPEL4WS files, centralizing the process within the repository may be more advantageous. In such cases, embedding processes with the registry represents an effective solution.

Table 1. Performance of External Document and Annotated UDDI Repository (milliseconds)

	Repo Size	*Add Serv. (ms)*	*Add Composite (ms)*	*Annotate (Note) (ms)*	*Collect Service IDs by Process (Aggregate) (ms)*	*Del. Serv. (ms)*
Ext. Process Doc	150	1573	2500	1500	1500	1900
	330	1573	2700	1700	1500	1900
	600	1573	2600	1600	1500	1900
	850	1573	2600	1600	1500	1900
Internal UDDI Data Structure	150	1573	2800	2100	2600	7790
	330	1573	3000	2100	4000	7790
	600	1573	3000	2100	5700	7790
	850	1573	3000	2100	7450	7790

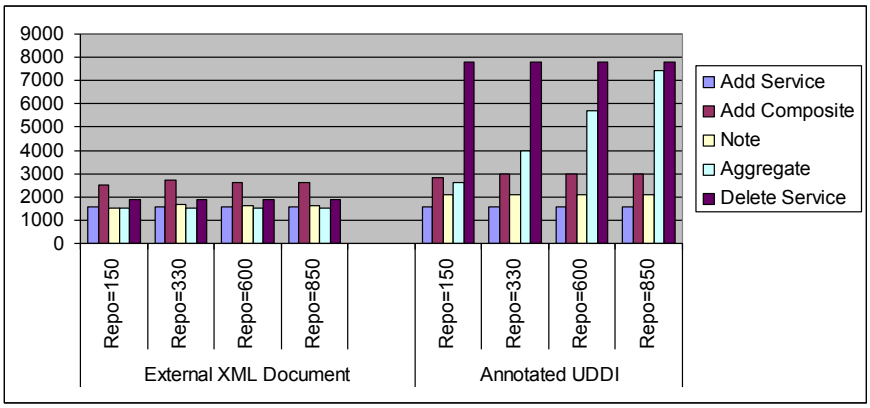

Figure 7. Comparison of Storing Process Information in External Document versus Annotating the UDDI Repository.

7. CONCLUSIONS

An innovation in this paper is the formalized model for web services-based business process and the relevant use cases for using this information. In addition, we introduce the design of a new interface for business-based UDDI interactions. Our experimentation evaluates the two leading approaches for capturing process information in UDDI registries. Overall performance information does not suggest a quantitative advantage for embedding process information directly into the repository. However, qualitatively, maintenance is less extensive since process information is centralized in a potentially federated registry. As future work, we plan to continue developing a process-oriented UDDI explorer and experiment on new approaches for interface design.

8. ACKNOWLEDGMENTS

We acknowledge fruitful conversations with Brian Schott and Robert Graybill of the University of Southern California, ISI-East and Suzy Tichenor of the Council of Competitiveness. This material is based on research sponsored by DARPA under agreement number FA8750-06-1-0240. This U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

9. REFERENCES

[1] Al-Masri, E. and Mahmoud, Q.H., "Crawling Multiple UDDI Business Registries", Proceedings of the 16th International Conference on the World Wide Web, Banff, Alberta, Canada, 2007

[2] Blake, M.B., Sliva, A.L., zur Muehlen, M., and Nickerson, J. "Binding Now or Binding Later: The Performance of UDDI Registries", IEEE Hawaii International Conference of System Sciences (HICSS-2007), Track on Technology and Strategies for Realizing Service-oriented Architectures with Web services, January 2007

[3] WS-BPEL(2008): http://www.ibm.com/developerworks/library/specification/ws-bpel/

[4] BPML (2008): http://www.ebpml.org/bpml.htm (currently moved to OMG)

[5] BPMN (2008): http://www.bpmn.org/

[6] Dogac, A., Tambag, Y., Pembecioglu, P, Pektas, S., Laleci, G., Gokhan, K., Toprak, S., and Kabak, Y. "An ebXML infrastructure implementation through UDDI registries and RosettaNet PIPs" Proceedings of the 2002 ACM SIGMOD Conference (SIGMOD 2002), Madison, Wisconsin, June 2002

[7] jUDDI (2008): http://ws.apache.org/juddi/

[8] Luo, J., Montrose, B., Kim, A., Khashnobish, A., Kang, M. "Adding OWL-S Support to. the Existing UDDI Infrastructure" Proceedings of the 4th International Conference on Web Services (ICWS2006), Chicago, Ill, November 2006.

[9] OWL-S(2008): http://www.daml.org/services/owl-s/

[10] Papazoglou, M. "Service-oriented computing: Concepts, characteristics and directions. In Proc. of WISE '03

[11] RDF (2008): http://www.w3.org/RDF/

[12] Sivashanmugam, K., Verma, K., and Sheth, A. Discovery of Web Services in a Federated Registry Environment, Proceedings of 4th IEEE International Conference on Web Services (ICWS), pp. 270-278, 2004.

[13] Spies, M., Schoning, H., and Swenson, K. "Publishing Interoperable Services and Processes in UDDI" The 11th Enterprise Computing Conference (EDOC 2007), Annapolis, MD, October 2007

[14] Srinivasan, N., Paolucci, M. and Sycara, K. "Adding OWL-S to UDDI, implementation and throughput," Proceedings of First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), San Diego, California, USA, 2004

[15] UDDI as the registry for ebXML Components, OASIS Technical Note, February 2004, Accessed (2008): http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-uddi-ebxml-20040219.htm

[16] Universal Description, Discovery, and Integration (UDDI) (2008): http://www.uddi.org/pubs/uddi_v3.htm

[17] Using BPEL4WS in UDDI Registry, OASIS Technical Note, July 2005 Accessed (2008): http://www.oasis-

open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-bpel-20040725.htm

[18] Zhang, L-J., Zhou, Q., and Chao, T., "A Dynamic Services Discovery Framework for Traversing Web Services Representation Chain", In Proceedings of the International Conference on Web Services (ICWS 2004), 2004

[19] Zhang, M., Cheng, Z., Zhao, Y. Huang, J.Z. Yinsheng, L., Zang, B. "ADDI: an agent-based extension to UDDI for supply chain management" Proceedings of the Ninth Int Conference on CSCW in Design, Shanghai, China, May 2005