

Taming Web Services in the Wild

Daniel R. Kahan
Dept. of Computer Science
Georgetown University
Washington, DC 20057
drk8@georgetown.edu

Michael F. Nowlan
Dept. of Computer Science
Georgetown University
Washington, DC 20057
mfn3@georgetown.edu

M. Brian Blake
Dept. of Computer Science
Georgetown University
Washington, DC 20057
blakeb@cs.georgetown.edu

Abstract

Service-oriented computing (SOC) enables organizations and individual users to discover openly-accessible capabilities realized as services over the Internet. Research in this area focuses on techniques for managing the messages that flow into and out of these services to ultimately compose higher-level functions. In our work, we investigate the nature of message definitions by analyzing real, fully-operational web services currently available on the Internet (i.e., from the wild). By leveraging insights into how real web service messages are defined, we develop enhanced syntactical methods to best aggregate these messages and ultimately the web services.

1. Introduction

Web services are networked capabilities with openly-accessible interfaces that can be discovered and executed in real-time [9]. Web service composition involves putting together a *chain* of multiple, related services. Using the semantic capabilities of technologies such as DAML [4], OWL-S [8], and BPEL4WS [3], web services messages can be correlated thus facilitating composition. This work investigates the use of enhanced syntactic approaches to facilitate composition. We believe that syntactic approaches can be used to aggregate services by their message names prior to semantic processing and, in effect, *tame* the target services repository. To create enhanced syntactic techniques, we analyzed real services in order to understand the actual *tendencies* of service developers.

2. Message naming tendencies in the Wild

We manually downloaded and verified the functionality of as many services as we could find on the Internet. Services came from a number of sources [1][11]. Two students spent approximately 40 hours downloading and verifying service functionality using Mindreef's SoapScope application [6]. From this effort,

we collected 490 WSDL documents. Table 1 lists the quantitative details of the repository.

Table 1. Detailed Information about the Repository

Statistic Description	Input	Output	Total
Number of Web Services			490
Gross Number of Part Names			12,187
Number of Part Names (Unique within each WSDL)	1,816	674	2,490
Overall Unique Part Names (23 names overlap inputs and outputs)	798	182	957

In order to scope our experiment, we decided to focus on the top 30 most common part names, which range from 536 occurrences to 5 occurrences of a particular part name. We loosely classified the top 30 part names as *ambiguous*, *descriptive*, or *more descriptive*. The part names are listed in Table 2. Figure 1 shows the breakdown of common part name classifications under our scheme.

Table 2. Top 30 Most Common Part Names.

Ambiguous	Descriptive	More Descriptive
Parameters, Body, Header LicenseInfo, Fault, Return Symbol, ResponseInfo, SubscriptionInfo, Result, Password, Identifier, Text IdentifierType, Type	LicenseKey, Username, Name, Height, Width, Style	StartDate, EndDate, Year, AsOfDate, City, Email, Country State, Month

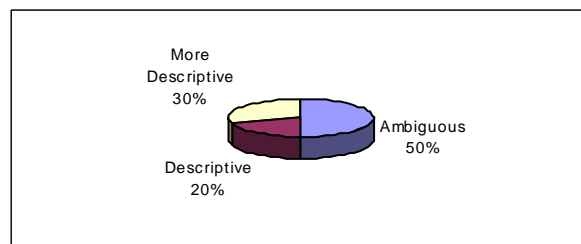


Figure 1. Percentage of Common Names by Type.

Ambiguous part names were common among services accessible via the Internet. We believe that these practices will be counter-productive to automated

approaches to both service discovery and composition as SOC increases in popularity.

3. Tendency-Based Syntactic Matching

We found several naming tendencies that are relevant for creating enhanced syntactic techniques.

- *Part names that are not common but that are similar to the common part names tend to be supersets or subsets of the common part name (for example; endDate is similar Date).*
- *Exceedingly lengthy strings and strings less than 2 characters are ineffective for message management.*
- *Setting a threshold for similarity distance is most effective when the threshold is not static but some function of the strings that are being compared.*

We introduce a matching algorithm called Tendency-Based Syntactic Matching-Levenshtein (TSM-L). This algorithm combines exploits the previously mentioned naming tendencies using Levenshtein distance (LD) (also called the *edit distance*). Levenshtein distance is a measure of similarity between two strings. In addition, excessively big or small strings are removed and subsumption relations evaluated.

4. Evaluation: Proactive Service Discovery

To evaluate the syntactical algorithms, we extracted specific strings from a travel itinerary document from Washington, D.C. to Orlando, Florida. We used those strings as the provided input parameters for a service discovery request. When the discovery request was executed on the entire repository using our software, 29 services were returned as results. A manual analysis of the results shows that at least 15 relevant services (true positives), 6 ambiguous services (possible false positives), and 9 false positives were discovered (Figure 12).

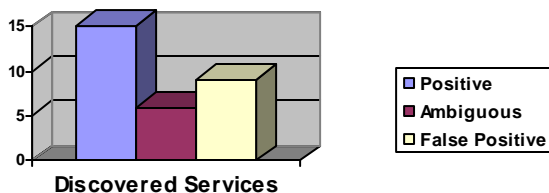


Figure 2. TSM-L Discovery Results on Repository.

TSM-L's ability to suggest 29 candidate services from a repository of 490 can significantly enhance future SOC routines. Leveraging more sophisticated

natural language approaches [2] to extracting the request would improve the accuracy of our approach.

5. Conclusion

In this work, we propose an approach that would facilitate semantic processing through enhanced syntactical matching. We introduce a new matching algorithm, TSM-L, which extends Levenshtein's algorithm by adding conditions that account for naming tendencies. The true innovation of our work is the bottom-up concept of incorporating human software development tendencies into automated string processing techniques.

6. Acknowledgement

The service repository and certain parts of the service discovery software used in this work were partially funded by the National Science Foundation under award number 0548514.

7. References

- [1] Amazon Web Services (2006): <http://aws.amazon.com/>
- [2] Bosca, A., Ferrato, A., Corno, D., Congui, I., and Valetto, G. "Composing Web Services on the Basis of Natural Language Requests", *Proceedings of the 3rd IEEE International Conference on Web Services (ICWS 2005)*, pp 817-818, Orlando, FL, June 2005
- [3] BPEL4WS (2006): <http://www.ibm.com/developerworks/library/ws-bpel>
- [4] DAML (2006): <http://www.daml.org/>
- [5] Merriam Park Software (2006): <http://www.merriampark.com/ld.htm>
- [6] Mindreef Soapscope (2006): <http://www.mindreef.com/>
- [7] NIST Levenshtein Distance (2006): <http://www.nist.gov/dads/HTML/Levenshtein.html>
- [8] OWL-S (2006): <http://www.daml.org/owl-s/>
- [9] Papazoglou, M. "Service-oriented computing: Concepts, characteristics and directions." *In Proceedings of WISE '03*
- [10] WS-Challenge (2006): <http://www.ws-challenge.org/>
- [11] XMethods (2006): <http://www.xmethods.com/>