Towards an Integrated Architecture for Composite Language Services and Multiple Linguistic Processing Components

Arif Bramantoro[1], Ulrich Schäfer[2], Toru Ishida[1]
[1]Department of Social Informatics, Kyoto University, Japan
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan
[2]Language Technology Lab, German Research Center for Artificial Intelligence, Germany
Campus D 3 1, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
E-mail: arif@ai.soc.i.kyoto-u.ac.jp, ulrich.schaefer@dfki.de, ishida@i.kyoto-u.ac.jp

Abstract

Web services are increasingly being used in the natural language processing community as a way to increase the interoperability amongst language resources. This paper extends our previous work on integrating two different platforms, i.e. Heart of Gold and Language Grid. The Language Grid is an infrastructure built on top of the Internet to provide distributed language services. Heart of Gold is known as middleware architecture for integrating deep and shallow natural language processing components. The new feature of the integrated architecture is the combination of composite language services in the Language Grid and the multiple linguistic processing components in Heart of Gold to provide a better quality of language resources available on the Web. Thus, language resources with different characteristics can be combined based on the concept of service oriented computing with different treatment for each combination. Having Heart of Gold fully integrated in the Language Grid environment would contribute to the heterogeneity of language services.

1. Introduction

One of the wide implementations of Web Services is language service (Shimohata, et al., 2001). The number of language service available on the Web is inevitably increasing. Computer scientists have been trying to develop more and more infrastructures to improve the quality and accuracy of the services. To utilize the language service more robustly, we need to integrate multiple infrastructures. Two of the famous ongoing developments of language infrastructures are the Language Grid (Ishida, 2006) and HoG (Heart of Gold; Schäfer, 2006).

The Language Grid is a framework of collective intelligence built on service oriented architecture which enables access to various language services and language resources in the world based on a single powerful protocol, HTTP. For the Language Grid, the more language resources it has the better it is for the availability of composite services. Composite language service means the ability to create a new service by combining existing services.

Heart of Gold (HoG) is also a framework that bridges user application and external natural language processing (NLP) components regardless the depth of the linguistic analysis. This framework provides integration between deep and shallow NLP annotations. Deep NLP applies as much linguistic knowledge as possible to analyze natural language sentences (Pollard & Sag, 1994). On the other hand, shallow NLP neglects the use of the whole range of linguistic details, but concentrates on specific aspects.

Only few shallow tools such as ChaSen and TreeTagger are provided by the Language Grid so far. There are various natural language processing (NLP) functions in HoG which are not provided by the Language Grid, especially the efficient deep analyzers for various languages. Moreover, hybrid and composite workflows can be defined that consist of combinations of the language components, the main goals being increased robustness and computation of formal semantics representations of natural language utterances.

This paper proposes an enhancement of the integrated architecture of the Language Grid and HoG that extends our previous work presented at the 2008 International Conference on Web Services (Bramantoro et al., 2008). Previously, the integrated architecture only provides HoG as an atomic service unable to be combined with other services in the Language Grid. Now, we utilize the composite language services in the Language Grid together with the multiple linguistic processing components in HoG.

The main contributions of this paper are (*i*) interoperability among various language services by creating new possible composition between multiple linguistic processing components of HoG and composite language services of the Language Grid; (*ii*) a new functionality of language services available on the Web by enabling the substitution of language components in HoG with additional in the Language Grid and vice versa within integrated composition.

2. Integrated Architecture

We identify three general problems concerning the integration.

- HoG is a framework based on components, while the

Language Grid is a service-oriented framework. We need to survey which architecture is suitable and reliable to accommodate these frameworks.

- The standard interfaces of these two frameworks are not the same. HoG provides XML annotations as output, while in the Language Grid standard interface there is no such type for output parameter.
- Both frameworks provide a processing strategy for language resources but in different ways. The Language Grid provides service workflows for composite language services, while HoG uses a compilable description language for composing multiple components.

To combine the two frameworks, a number of experiments were designed to combine HoG and the Language Grid. We found out that the best possible one for combining HoG and the Language Grid is by wrapping HoG as a Web service that can be accessed through the Language Grid. We proposed that the Language Grid can utilize HoG by adding it to the language resources layer, a layer where atomic services are wrapped and registered. Although it is not common in the Language Grid to have a composite service in this layer, the standard wrapping technique of the Language Grid requires doing so. Consequently, we have to treat HoG differently in this layer since it contains multiple NLP components that behave as composite services.

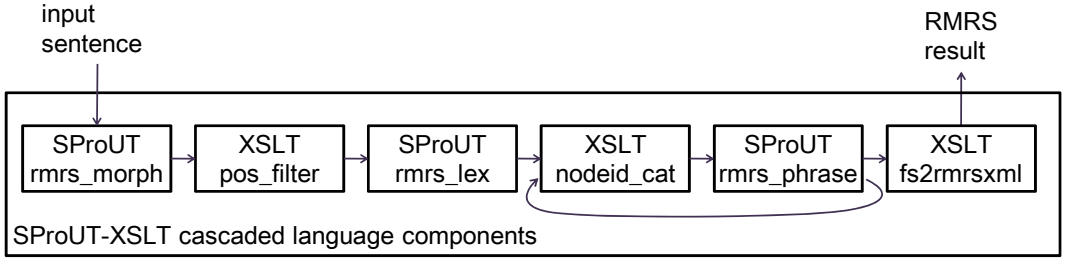
We create a new Web service that can connect to HoG and implement the Language Grid standard interface. From HoG's point of view, this Web service acts as an application, whilst from the Language Grid's point of view, this Web service is considered as a wrapped language resource. The wrapped Web service connects to the Module Communication Manager via XML RPC. Therefore, the HoG server can be located at any nodes in the Language Grid.

3. Processing Flow and Workflow

To get a higher quality of language processing we need to integrate more than one processing tool. HoG allows the user to execute more than one language component. In fact, this multiple component processing is the original characteristic of HoG since the default strategy is to execute the shallowest component first, then other components with increasing depth up to the requested depth. Unless a user defines smallest depth value, there is more than one language component executed.

There are three ways to configure the sequence of the components in HoG, (1) varying the depth value, (2) varying input and output, (3) using the SDL extension. In this paper, we focus on using SDL extension for running multiple components in a HoG service integrated in the Language Grid. It is impractical to implement the concept of depth value in service oriented computing. Moreover, Web services should be autonomous so that it is difficult to vary the input and output of language services during the composition.

SDL (System Description Language; Krieger, 2003), is a specific language initially used for building NLP systems and may be used in HoG to define sub-architectures of composite components. SDL uses a declarative specification language to define a flow of information (input and output) between linguistic processing components. The declarative specification consists of operators, symbolic module names, assignment of these symbolic module names to Java class names and constructor arguments. The basic operators currently available in HoG are + (*sequence*), | (*parallelism*), and * (*unrestricted iteration*). For example, multiple linguistic components consist of three SProUT grammar components and three XSLT transformation components described in Figure 1 together with its definition in SDL syntax.




```
chunkiermrs = ( sprout_rmrs_morph + xslt_pos_filter + sprout_rmrs_lex +
               (* xslt_nodeid_cat + sprout_rmrs_phrase ) + slt_fs2rmrsxml)

sprout_rmrs_morph = SproutModulesTextDom("rmrs-morph.cfg")
xslt_pos_filter = XsltModulesDomDom("posfilter.xsl", "aid", "Chunkie")
sprout_rmrs_lex = SproutModulesDomDom("rmrs-lex.cfg")
xslt_nodeid_cat = XsltModulesDomDom("nodeinfo.xsl", "aid", "Chunkie")
sprout_rmrs_phrase = SproutModulesDomDom("rmrs-phrase.cfg")
xslt_fs2rmrsxml = XsltModulesDomDom("fs2rmrsxml.xsl")
```

Figure 1: Composing NLP components in Heart of Gold with SDL

Composite services in the Language Grid are formulized in constraint satisfaction problem specification (Bramantoro & Ishida, 2009). Constraint satisfaction problem adopted from artificial intelligence theory is characterized with triplet entities (X, D, C) as follows:

- $X=\{X_1,\ldots,X_n\}$ is a set of abstract Web services, with $X_i.IN$ is a set of required input types, $X_i.OUT$ is a set of required output types, $X_i.QOS$ is a set of required QoS types. These requirements are defined as abstract service specifications..
- $D=\{D_1,\ldots,D_n\}$ where D_i a set of concrete Web services X_i that can perform the task of the corresponding abstract Web services.
  $D_i=\{s_{i1},...,s_{ik}\}$ where s_{ij} is a concrete Web service of the corresponding X_i with $s_{ij}.IN$ is a set of provided input types, and $s_{ij}.OUT$ is a set of provided output types, $s_{ij}.QOS$ is a set of provided QoS types. In semantic matching of web service (Paolucci et al., 2002), every element of the input set in concrete service specification should be also an element of the input set in abstract service specification and every element of the output set in abstract service specification should be also an element of the output set in concrete service specification. We argue that in QoS based matching every element of the QoS set in abstract service specification should be also an element of the output set in concrete service specification. Therefore, we define semantically matched service specification as follows.
  $D_i=\{s_{ij} \mid s_{ij}.IN \subseteq X_i.IN \wedge X_i.OUT \subseteq s_{ij}.OUT \wedge X_i.QOS \subseteq s_{ij}.QOS\}$
- $C=\{C_1,\ldots,C_p\}$ is a set of constraints which consists of workflow control, QoS-related, provider-defined and user-defined constraints.

In the Web service composition, there are four possible controls of workflow, i.e. *sequence*, *split*, *choice* and *loop* that can be specified in a constraint satisfaction problem. For example, in order to increase the quality of translation, we can compose a translation service with the community dictionary service in the Language Grid as described in Figure 2.

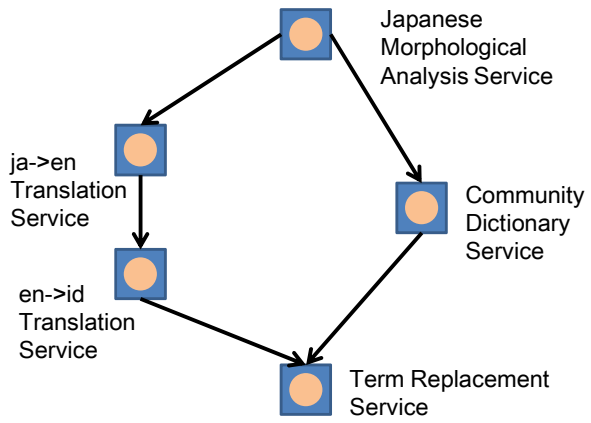


Figure 2: A workflow of specialized translation service between Japanese and Indonesian

The formulization for this workflow is as follows:

- $X=\{X_1, X_2, X_3, X_4, X_5\}$, where:
  - X_1: Morphological analyzer service;
  - X_2: ja-en translation service;
  - X_3: en-id translation service;
  - X_4: Community dictionary service;
  - X_5: Term replacement service;
- $D=\{D_1, D_2, D_3, D_4, D_5\}$, where (for the sake of simplicity, we omit the input and output parameters of D_i)
  - D_1: {mecab at NTT, ICTCLAS, KLT at Kookmin University, treetagger at IMS Stuttgart};
  - D_2: {JServer at Kyoto-U, JServer at NICT, WEB-Transer at Kyoto-U, WEB-Transer at NICT};
  - D_3 : {ToggleText at Kyoto-U, ToggleText at NICT};
  - D_4: {Science Dictionary, Natural Disasters Dictionary, Tourism Dictionary at NICT, Academic Terms Dictionary at NII};
  - D_5: {TermRepl service};
- C including (due to page limitation, only example constraints are shown)
  - C_1: For multi hop translation, $X_2.OUT=X_3.IN$;
  - C_2: For composite service which involves X_2 and X_4 (translation service and multilingual dictionary), serverLocation(X_2)=serverLocation(X_4);
  - C_3: For morphological analysis used together with community dictionary services, partialAnalyzedResult($X_1.OUT$) $\in$ $X_4.IN$.

4. Combination of Two Flows

There are two urgent combinations between the multiple linguistic processing components of HoG service and composite language services in the Language Grid. These combinations involve the processing flow of HoG service and the workflow of the Language Grid.

Firstly, we need to incorporate composite components of HoG into the Language Grid's workflow. For example, there is a specialized Japanese-English translation service in the Language Grid that includes a Japanese morphological analyzer, an English morphological analyzer and some community dictionary services. The concrete Web service for English morphological analyzer available in the Language Gird is TreeTagger.

Multiple linguistic processing components (TreeTagger and RMRS) in HoG provide not only morphological analysis but also named entity recognition. This new functionality in the Language Grid's workflow enables users to dynamically select the right community dictionary service during workflow execution. Therefore, we can substitute the English morphological analyzer service in the workflow with the ones from HoG. To realize this combination, we have to instrument a new Web service in the workflow, i.e. an XML decoding service to detach the XML code in the HoG service output.

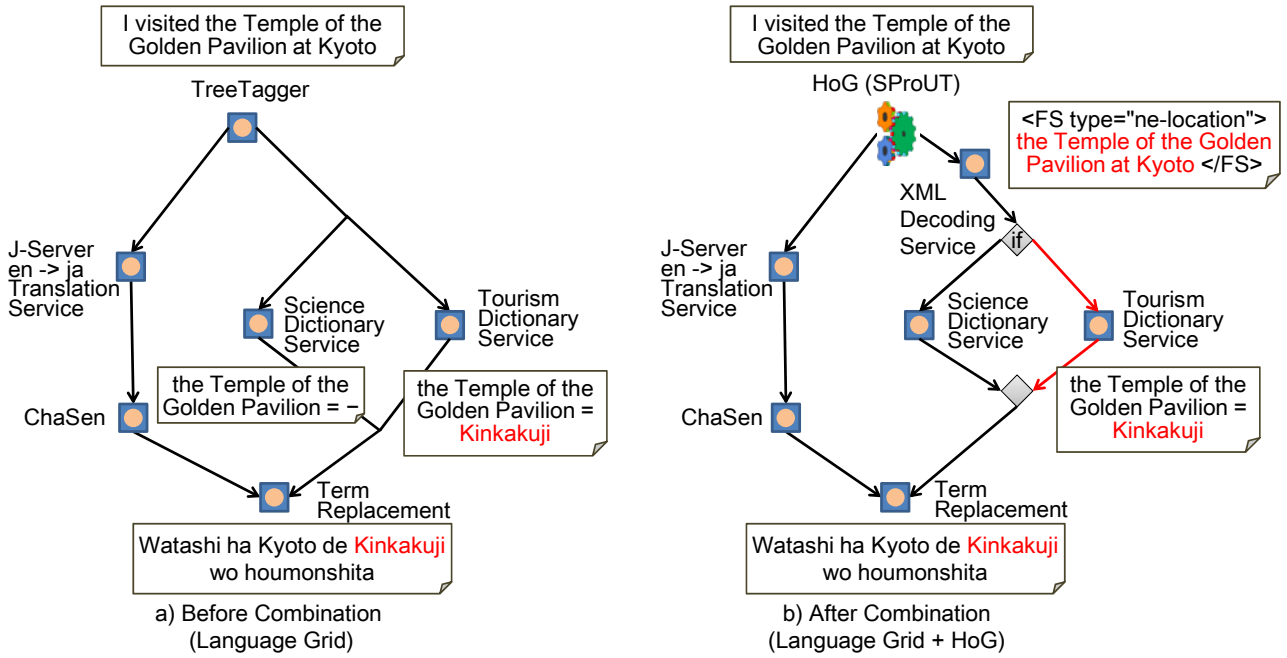


Figure 3: HoG composite components in the Language Grid's workflow

Figure 3 shows the scenario of combining HoG service in the Language Grid's workflow. In this scenario, a location term in the sentence could be detected and tagged by named entity recognition component (SProUT). When the location term is tagged by SProUT, the workflow execution engine automatically chooses Tourism Dictionary Service instead of Science Dictionary Service. The final result is the same as the existing workflow before combination, but the workflow execution by using HoG service should be more efficient since it runs one dictionary service in one time, not all dictionaries in parallel.

The scenario of using HoG service in the Language Grid workflow is also applicable to other dictionary services in the Language Grid. This could be realized by using the current tag set in the named entity recognition component related to the dictionary service or training a new tag set according to dictionary service entries. The integration will deliver efficiency since most of the community dictionary services are not free. Currently, there are more than 15 dictionary services available in the language grid. It should be costly to run all community dictionary services in each workflow without utilizing HoG service.

Secondly, we need to incorporate language service(s) of the Language Grid inside the processing flow of HoG. To do this, it is necessary to realize a mechanism of Service as a Software (SaaS) by wrapping language service(s) in the Language Grid as a HoG component that has additional parameters of XML output and, therefore, needs a special tool to convert the service output into XML format.

This integration is useful when we want to try the NLP components of HoG in different languages. For example, ChunkieRMRS in HoG is only available in German and English. Hence, deep NLP for Japanese could also be realized by utilizing Japanese-English translation service from the Language Grid (it is important to note that composite language service such as multi-hop translation service can be also wrapped as a language component) as described in Figure 4.

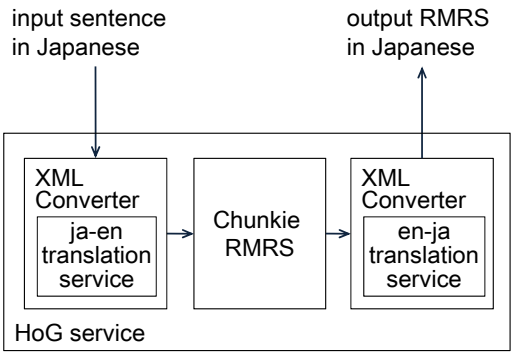


Figure 4: Language service inside HoG's processing flow

To realize the combinations, we propose a service and its architecture to integrate the processing flow and workflow. This service consists of processing flow analyzer, workflow analyzer and SDL writer. Three repositories are utilized by this service, i.e. language component information, language service information and extended workflow repository represented in constraint satisfaction problem.

An alternative workflow is automatically created and stored in the workflow repository together with its generated SDL description of incorporated HoG's components. When a user requests a particular task to be

performed by composite language services, the processing flow & workflow integrator service analyzes an alternative workflow, enriches it with deeper composite language components provided by the HoG service, and calls SDL Writer to generate a new SDL description based on a new workflow combination to be delivered to the user. In addition, this integrator service can run offline so that the processing time of a user request is not affected since the new workflow has already been stored in the repository before runtime. The overall service architecture is illustrated in Figure 5.

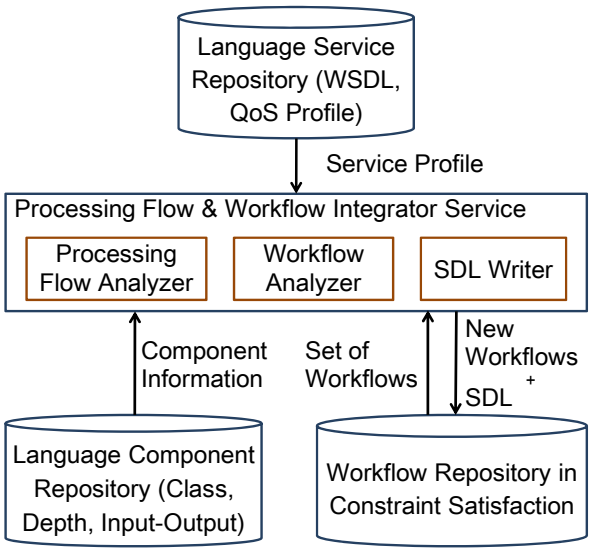


Figure 5: Integrator service architecture for composite language services and components

5. Related Work

We realize that there have been some breakthroughs in NLP researches that try to transform language software components into more loosely coupled components by using standard internet technology so called Web services. However, it is hard to find a good reference that provides a real solution for a complex integration task between a huge web service framework (the Language Grid) and a dynamic, highly customizable software system such as HoG.

Today's era is service oriented computing that creates everything as a service. There are many considerations to be examined before transforming software into a service. We can accommodate all language resources as a service but converting individual resources takes a lot of efforts as in the Language Grid. It is much easier to convert an existing platform that contains multiple language resources. Then, one would still be able to intervene inside the platform to choreograph individual resources.

A hybrid approach proposed by Jang et al. (2004) provides a workflow architecture based on Web services and object-oriented techniques. The authors argue that this architecture supports workflow systems with multiple process languages and standardized resource management. An interesting idea of this paper is the ability to support different web service-supporting process definition languages, such as BPML, XPDL, BPEL, and WSCI. This idea has been inspiring us to have different description languages in a single architecture. However, this paper only provides a few explanations on the implemented prototype.

A similar effort has been proposed in W3C to deal with different types of web services. Kavantzas et al. (2005) propose WS-CDL (Web Service Choreography Description Language) that is mainly used to integrate several web services from different providers, implementing different Web service technologies, such WS-BPEL and .Net C#. More specifically, WS-CDL supports the interoperability and interactions between web services in various programming languages and platforms within one business function by optimizing messaging between web services. This situation is different from what we face in the language domain. The Language Grid uses constraint satisfaction for its composite services. The HoG service is integrated into the Language Grid at a language resource layer (considered as atomic service), but contains composite components within its processing flow in SDL. Problems faced during the integration are not related to messaging between web services but mostly lie in transforming existing multiple linguistic processing components into machine-readable composite web services.

There is another candidate recommendation by W3C to define a new language, XProc (XML Processing Language; Walsh et al., 2009), to compose XML processes and deal with operations to be performed on XML documents. One of the advantages of this language is that it supports HTTP requests. By using this feature, this specification might be useful to integrate language services defined in WSDL and SOAP (both use XML over HTTP) and language components with XML output and called by XML-RPC. A specific pipeline can be created to process composite language services and multiple linguistic processing components at the same time. The concept of XProc is suitable to integrate two XML-based architectures, but currently there is no guarantee that XProc can fully support language services, especially for language services which are not merely an XML document.

Another open platform for natural language processing, Unstructured Information Management Architecture (UIMA) developed by IBM researchers (Ferrucci & Lally, 2004), enables association of each element of an unstructured document with semantic results of analysis. This paradigm can be adapted to the Language Grid. Any word in the source text translated by the Language Grid can be initially assigned a semantic value from UIMA. To give a simple example, the word "car" in a text document can be associated with multiple analysis engines, e.g. a

morphological analysis and a translation engine. The result would be the word “car” with associated semantic values “noun:en” and “kuruma: en → ja”. These associations could be further processed by more advanced language-aware applications. Having two frameworks, HoG and UIMA, in the Language Grid could be another research topic, taking into account considerations on HoG and UIMA integration discussed in Schäfer (2008).

6. Conclusion

In this paper, we showed that language resources with different characteristic can be combined based on the concept of service oriented computing with different combinations. Multiple linguistic processing components in HoG can be combined with the existing workflow of composite services in the Language Grid environment. On the other hand, the composite language services in the Language Grid can be utilized in the processing flow of HoG components.

The next step that can be done on the basis of this prototype is to build more applications for visualizing computed annotation results. Currently, the return value of HoG service is an XML document, which is complicated for layman to understand and use. By providing client applications that process and visualize the XML result, the users of the Language Grid, not only linguists, could hopefully benefit better from natural language processing results returned by HoG.

7. Acknowledgements

This research was partially supported by Strategic Information and Communications R&D Promotion Programme from Ministry of Internal Affairs and Communications, and also from Global COE Program on Informatics Education and Research Center for Knowledge-Circulating Society.

The work described in this paper was partially supported by the German Federal Ministry of Education and Research under contract 01IW08003 (project TAKE: Technologies for Advanced Knowledge Extraction).

8. References

Bramantoro, A. & Ishida, T. (2009). User-Centered QoS in Combining Web Services for Interactive Domain, In *Proceedings of the International Conference on Semantics*, Knowledge and Grid, Zhuhai, China, October 2009, pp. 41-48.

Bramantoro, A., Tanaka, M., Murakami, Y., Schäfer, U., & Ishida, T. (2008). A Hybrid Integrated Architecture for Language Service Composition, In *Proceedings of the IEEE International Conference on Web Services*, Beijing, China, September 2008, pp. 345-352.

Ferrucci, D. & Lally, A. (2004). Building an Example Application with the Unstructured Information Management Architecture, *IBM Systems Journal*, 43(3), pp. 455–475.

Ishida, T. (2006). Language Grid: An Infrastructure for Intercultural Collaboration, In *Proceedings of the IEEE/IPSJ Symposium on Applications and the Internet*, Arizona, USA, January 2006, pp. 96-100.

Jang, J., Choi, Y., Zhao, J.L. (2004). An Extensible Workflow Architecture through Web Services, *International Journal of Web Services Research*, 1(2), pp. 1-15.

Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., & Barreto, C. (2005). Web Service Choreography Description Language (WS-CDL) Version 1.0, W3C Candidate Recommendation, World Wide Web Consortium. Retrieved November 9, 2009, from http://www.w3.org/TR/ws-cdl-10.

Krieger, H.-U. (2003). SDL—A Description Language for Building NLP Systems, In *Proceedings of the HLT-NAACL Workshop on the Software Engineering and Architecture of Language Technology Systems*, Edmonton, Canada, May 2003, pp. 84–91.

Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K. (2002). Semantic Matching of Web Services Capabilities, In *Proceedings of the International Semantic Web Conference*, Sardinia, Italy, pp. 333-347.

Pollard, C. J. & Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*, University of Chicago Press.

Schäfer, U. (2006). Middleware for Creating and Combining Multi-dimensional NLP Markup. In *Proceedings of the EACL-2006 Workshop on Multi-Dimensional Markup in Natural Language Processing*. Trento, Italy, April 2006, pp. 81–84.

Schäfer, U. (2008). Shallow, Deep and Hybrid Processing with UIMA and Heart of Gold, In *Proceedings of the LREC-2008 Workshop Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP*, Marrakesh, Morocco, May 2008, pp. 43-50.

Shimohata, S., Kitamura, M., Sukehiro, T., & Murata, T. (2001). Collaborative Translation Environment on the Web, In *Proceedings of the Machine Translation Summit VIII*, Santiago de Compostela, Spain, September 2001, pp. 331-334.

Walsh, N., Milowski, A., & Ritzinger, S. T. (2009). XProc: An XML Pipeline Language, W3C Candidate Recommendation, World Wide Web Consortium. Retrieved December 7, 2009, from http://www.w3.org/TR/xproc/.