

A Hybrid Integrated Architecture for Language Service Composition

Arif Bramantoro¹, Masahiro Tanaka¹, Yohei Murakami², Ulrich Schäfer³, Toru Ishida^{1,2}

¹*Department of Social Informatics, Kyoto University, Japan*

{arif, mtanaka}@ai.soc.i.kyoto-u.ac.jp

ishida@i.kyoto-u.ac.jp

²*Language Grid Project, National Institute of Information and Communication Technology, Japan*

yohei@nict.go.jp

³*Language Technology Lab, German Research Center for Artificial Intelligence (DFKI),*

Saarbruecken, Germany

ulrich.schaefer@dfki.de

Abstract

This paper reports on our experiences with combining Heart of Gold and Language Grid technology to provide more language resources available on Web. Heart of Gold is known as middleware architecture for integrating deep and shallow Natural Language Processing components. The Language Grid is an infrastructure built on top of the Internet to provide distributed language services. Having Heart of Gold available as Web services in the Language Grid environment would contribute to interoperability among language services.

1. Introduction

The Web initially only stored and distributed static data, such as text and images. Today's Web has evolved into a provider of services, known as Web Services. These include information-providing services such as weather information, stock quotes, flight information, and external world-modifying services such as dynamic weather warning, stock sales, and flight reservation.

One of the wide implementations of Web Services is language service [1,10]. 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 development of language infrastructures are Language Grid [2] and Heart of Gold (HoG) [3].

The Language Grid is a framework which tries to collect various language services and language resources in the world within single powerful protocol, i.e. 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 tries to accommodate any Natural Language Processing (NLP)

tool regardless the depth of the analysis. The return value of this framework is an XML standoff annotation string which can be further processed by any applications. Since HoG provides a high degree of flexibilities for accessing its server as well as defining processing strategies for integrating multiple NLP modules, many new applications have been developed utilizing HoG to gain the benefits of such hybrid analysis results [4].

The language services for Natural Language Processing (NLP) provided by the Language Grid are similar to NLP functions provided by HoG. However, only a few shallow tools such as ChaSen and TreeTagger are provided by the Language Grid so far. There are various NLP functions in HoG which are not provided by the Language Grid, especially the efficient deep parser PET, but also shallow tools such as tokenizers, part-of-speech taggers, morphological analyzers, named entity recognizers, PCFG and dependency parsers, sentence boundary detectors and coreference resolvers, accompanied by linguistic resources for various languages. Moreover, hybrid and composite workflows can be defined that consist of combinations of the NLP components, the main goals being increased robustness and computation of formal semantics representations of natural language utterances.

On the other hand, functionalities for management of access to language resources are implemented for the Language Grid, but not for HoG. Companies and research organizations which provide their language resources for the frameworks usually have conditions for use in order to protect the intellectual property rights of their language resources. Therefore, access to language resources must be controlled. Moreover, statistics of access to resources should be available to service providers.

As described above, the Language Grid and HoG can complement each other. Therefore integrating them into single framework will have great benefit for users and service providers of both of the two frameworks.

We have identified two 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.

This paper proposes that the Language Grid can be enhanced through the integration with HoG. Therefore there will be more language resources available in the Language Grid environment. In order to fulfill this scenario, HoG should be wrapped as a Web service that implements the Language Grid interface. In this way, we believe that HoG can be accessed through the Language Grid.

In the next section, we will briefly discuss HoG and the Language Grid. Then the proposed hybrid architecture is presented followed by a use scenario that utilizes HoG service. The last section would be conclusion and future work.

2. Two Architectures for Composite Language Services

In this paper, we will not discuss the Language Grid and HoG in detail. Details of the Language Grid are available at its trilogy websites, i.e. Language Grid association¹, Language Grid operation center² and NICT Language Grid Project³, whereas details of HoG⁴ are available in [4].

2.1. Language Grid: Web Service Composition

The Language Grid is an infrastructure for coordinating distributed language services on the Web. Various services such as machine translators, morphological analyzers and bilingual dictionaries and parallel texts are available on the Language Grid. The services that provide the functionality of a single language resource are called atomic services.

On the other hand, composite services which are composed of multiple services are described in WS-BPEL [6] and deployed on the Language Grid. For example, a composite service cascades multiple machine translators in order to support language pairs which any single machine translator does not support. Another composite service is composed of a machine translator, a

morphological analyzer and a technical term dictionary in order to translate sentences in a specialized domain.

The interfaces of all services on the Language Grid are designed based on a language service ontology [7] and standardized according to the types of the services. This makes the differences between specifications of language resources by various vendors transparent.

We note that the language resources available on the Language Grid are provided by a wide variety of companies, research organizations and individuals and that issue on the intellectual properties must be considered [15]. Language resource providers impose conditions on the use of their own resources. For example, machine translators are usually developed by for-profit companies and cannot be provided without charge. To use the services, a contract on the limitation of use, such as users, expiration date and number of usage, has to be signed. Most openly available morphological analyzers have been developed by research organizations. In many cases, they allow only non-profit use. However, different terms of use may apply when their resource is published on Web.

The Language Grid has functionalities for access control to deal with the issues on intellectual rights of language resources, and consists of two kinds of nodes; core nodes and service nodes. Core nodes take charge of a repository and management of services available as part of the Language Grid.

Language resources are deployed on service nodes and accessed from core nodes. A client application first sends a request to a core node. The core node next finds the required services searching its repository or inquiring other core nodes. Then the core node invokes services found on service nodes. During this process, the core node authorizes the client in order to confirm the access rights and the number of invocation which the client is allowed. The language resource provider can configure the permission and monitor the statistics of use of his/her resource anytime.

Table 1. Language resources in the Language Grid

No	Category	Resources	Language
1	machine translator	Netat, Petra, JServer, Sdlfretr, WEB-Transer	ja,en,ko,zh-cn,de,fr,it,es,pt
2	parallel text	Medical scenes, Kawasaki city	ja,en,ko,zh,es
3	morphological analyzer,	KLT, MeCab, TreeTagger, ChaSen	ja,en,ko,zh,es,de,fr,it,nl,ru,bg
4	bilingual dictionaries	Picton, EDR Japanese-English	ja,en

Currently, there are about 70 language resources published in the Language Grid [8]. These resources are categorized into machine translator, parallel text, morphological analyzer, bilingual dictionaries and adjacency pair. A list of several resources currently

¹ <http://www.langrid.org/association/indexe.html>

² <http://langrid.org/operation/en/>

³ <http://langrid.nict.go.jp/en/>

⁴ <http://heartofgold.dfki.de>

available in the Language Grid can be seen in Table 1. Not all language resources support all listed languages, nor all language combinations supported⁵. The underlined resources will be discussed in the next section.

The Language Grid has mainly focused on standalone language services such as machine translators and dictionaries for many languages so far. The current available language services cover several languages: Japanese, Chinese, Korean, English, German, Spanish, French, Italian, Dutch, Russian, and Bulgarian and so on. However, the possible process on the Language Grid is still limited because of the lack of services which analyze an expression in a natural language in detail. Therefore shallow or deep natural language processing services will enhance the process on the Language Grid.

2.2. Heart of Gold: Functional Composition

Basically, Heart of Gold provides integration between deep and shallow Natural Language Processing (NLP) annotations. Deep NLP applies much linguistic knowledge as possible to analyze natural language sentences [14]. On the other hand, shallow NLP neglects the use of the whole range of linguistic details, but concentrates on specific aspects. Both shallow and deep NLP may involve statistical as well as rule-based methods for linguistic modeling. It could be shown that combining shallow and deep NLP can improve the analysis results in terms of accuracy and efficiency [4].

In HoG, there are two kinds of interfaces. The first interface is between applications and HoG. This interface enables application to communicate either via Java API (for Java applications) or XML-RPC (for remote applications or applications written in other programming languages). To open a communication to HoG in this interface, an application creates an instance of the HoG architecture.

The second interface is between NLP components and HoG. In this interface, the components are wrapped as subclasses of either Module (local Java-based components) or XmlRpcModule (remote, possibly non-Java, components).

In HoG, the default processing strategy is shallowest component first (e.g. tokenizer), then other components with increasing depth, up to the requested depth. It will return to the result of the previous component if there is no result from the component with requested depth. Each component gets the output of previous component as input plus the output from other components if configured. The result of the query is the result of the deepest component in sequence. Analysis results from previous components are returned on request. Processing

order and information flow between modules can be configured by varying parameters such as module depth, additional input or output annotation. Even more flexible processing strategies, i.e. parallelism and loops, are supported through the SDL extension.

The detail list of NLP tools currently provided by HoG can be seen in Table 2. These tools support some languages individually⁶ and various NLP tasks, such as tokenization, morphological analysis, part of speech tagging, named entity recognition, statistical parsing, coreference resolution, and deep parsing. The tools with underlined text in the table are also provided by the Language Grid. The rests of the tools in the table are new to the Language Grid. They could be useful for future applications in the Language Grid.

Table 2. NLP tools provided by HoG (details and references can be found in [4] and on the Heart of Gold website)

No	Name	Description	Language
1	JTok	tokenizer	en,de,it
2	<u>ChaSen</u>	<u>morph</u>	<u>ja</u>
3	TnT	PoS tagger	en,de
4	<u>TreeTagger</u>	<u>PoS tagger</u>	<u>en,de,es,it</u>
5	FreeLing	PoS, morph, NE	en,es,ca,gl,it
6	Chunkie	chunker	en,de
7	ChunkieRmrs	chunks	en,de
8	SProUT	morph, NER, IE	en,de,el,ja
9	LingPipe	NER	en
10	Corcy	coref. resolver	en
11	RASP	stat. parser	en
12	SDL	subarchitectures	
13	Sleepy	shallow parser	de
14	PET	deep parser	en,de,el,it,ja,no
15	RMRSmerge	merge RMRSes	

The main advantage of using HoG is the availability of deep NLP for computing deep semantics representations for natural language sentences (sentence semantics). For example, as mentioned in [4], when we receive a sentence “John gave Mary a book” then we can have a deep analyzed result as *past(give(John, Mary, book))*. A further advantage of HoG is that it optionally supports a common, uniform semantics representation format, RMRS [13], for both shallow and deep component results. The idea is basically that shallow analysis generates underspecified semantics representation compared to results available from deep parsing on the same input. This semantics representation then can be utilized in any linguistic application, such as relation extraction, text summarization, question answering, opinion detection, or command interpretation.

⁵ ISO language codes: ja=Japanese, en=English, ko=Korean, zh-cn=Chinese, ko=Korean, de=German, fr=French, it=Italian, nl=Dutch, es=Spanish, pt=Portuguese, ru=Russian, bg=Bulgarian

⁶ Additional language codes: ca=Catalan, gl=Galician, el=Greek, no=Norwegian

3. Hybrid Architecture for Composite Language Service

To combine the two frameworks, the Language Grid and Heart of Gold, it is necessary to find the most useful and reliable scenarios. A number of alternatives were designed to combine HoG and the Language Grid. Because of the limited space in this paper, we will not present the alternative architectures here. We found out that the most possible scenario for combining HoG and the Language Grid will be by wrapping HoG as a Web service, so that it can be accessed through the Language Grid.

3.1. Language Grid / Heart of Gold

Here are the characteristics of HoG that can be considered for realizing a Web service on top of it:

1. Many applications such as question answering, text summarization, information extraction, and machine translation utilize HoG as their NLP analysis tool. Therefore, there is strong evidence that other, Language-Grid-based application would benefit, too.
2. HoG is an XML-based architecture. Moreover, one of the methods to communicate with its managing module (MoCoMan), either from application or external modules, is XML-RPC. With XML-RPC, any application or module can get easily connected to HoG even though remotely located on different server.
3. HoG can provide flexible processing strategies through processing order and information flow, between NLP modules including parallelism and loops. This is comparable to composite Web services [9].

Based on the above mentioned characteristics, the proposed architecture for HoG and the Language Grid integration is illustrated in Figure 1. The left part of Figure 1 shows the architecture of the Language Grid, while the right part shows that of HoG. The architecture of the Language Grid consists of four layers, P2P Grid Infrastructure, Language Resources, Language Services, and Intercultural Collaboration Tools.

The first layer provides the mechanism of information sharing among core nodes. Language resources are wrapped as atomic Web services and registered at the second layer. Composite services consisting of atomic services are published at the third layer. Tools for intercultural collaboration, such as a multi-lingual chat, are provided at the last layer and invoke the services at the third layer. The Language Grid can utilize HoG by adding it to the Language Resources layer.

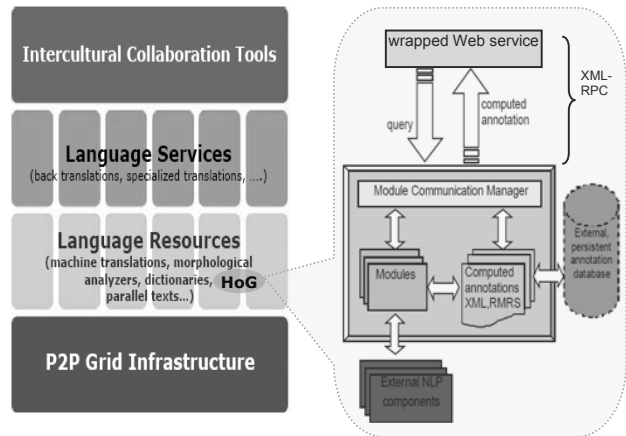


Figure 1. Wrapping architecture

To realize this scheme, we create a new Web service that can connect to HoG. HoG is then wrapped as a Web service that implements 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 will connect to the Module Communication Manager (MoCoMan) via XML RPC. Therefore, the HoG server can be located at a different node in the Language Grid.

To see in detail how we combine HoG and the Language Grid, we show in the class diagram in Figure 2 the prototype combining HoG with the Language Grid. There is a new Web service in the Language Grid service node, which is currently located in the package of `jp.go.nict.langrid.wrapper.hog`. This Web service inherits and implements two upper classes, `LocalHoGService` and `AbstractHoGService`, to provide a standardized Language Grid interface. We remove `HogXmlRpcClient` class from HoG server and put it in this package. The `HogXmlRpcClient` class will communicate with the HoG server to send a request and receive the analyzed result via XML-RPC.

The implementation of the above architecture is as follows:

1. To adapt the Language Grid standard, there are two additional created classes, i.e. `AbstractHoGService` and `LocalHoGService`.
2. The service of HoG itself is in the `HoGService` class which implements `AbstractHoGService` and uses `LocalHoGService`. The `HoGService` class overrides the method of `AbstractHoGService` (`doAnalyze`) by calling `LocalHoGService`'s method.
3. `AbstractHoGService` contains an interface method for accessing the service. This interface method, `analyze`, invokes the protected abstract method `doAnalyze`.

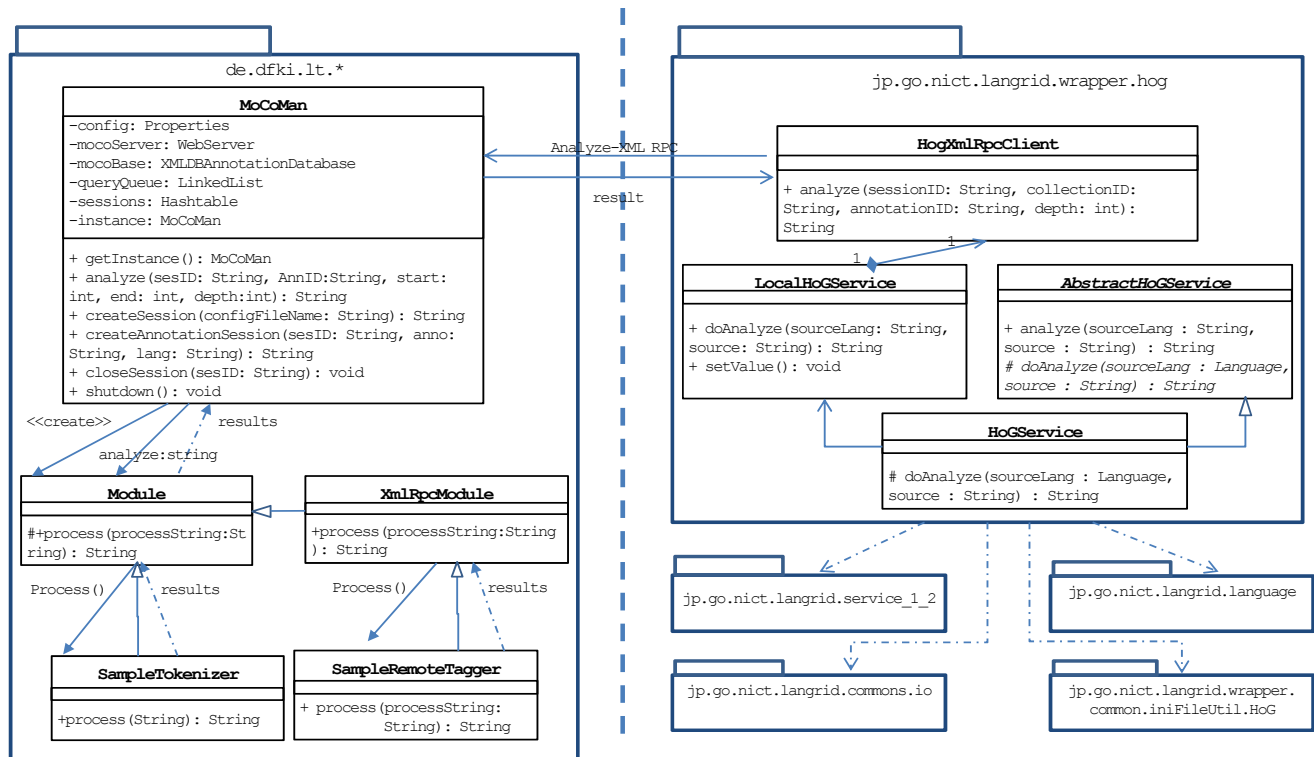


Figure 2. UML class diagram of integration architecture

4. LocalHoGService contains a client method of the services. This class uses HogXmlRpcClient for connecting to the HoGServer.

Access to a NLP service of HoG in the Language Grid works in the same way as access to other services in the Language Grid. Basically, getting access to a HoG service is simply grabbing the WSDL file by appending the string `?wsdl` to the end of HoG URL in `http://hog.nict.go.jp/axis/services/HoG?wsdl`. This service description file can be stubbed to create a client program. The created client program can use the `analyze` function of the HoG service to get annotation results from the HoG server. The result of this Web service function is an XML string containing the annotation result from the selected NLP modules. The return value of this service is contained in a string, which is not standard in the Language Grid. Therefore, we modify the infrastructure of the Language Grid so that there will a new standard for HoG which returns the XML annotation string as a service output.

An XML string is easily handled by using XPath⁷ in a composite service. By doing this, it is possible for other services in the composite service to utilize the result of analysis returned by the HoG service.

While Web services are language independent, there are two main preferences for Web service programming platforms, Microsoft's multi-language .NET and Sun's J2EE framework [11]. Both platforms can be used to create a client program accessing this HoG service.

To access the reliability of the new services, two kinds of test have been conducted:

- a. JUnit local test
This test examined the functionality of the service by creating an instance of HoGService class and running it as JUnit test. In this test, HoGService worked well and returned annotated XML string of NLP analysis.
- b. Client Web service test
This test examined the functionality of the service via service URL. To conduct this test, we generated a client stub by utilizing the Apache Axis tool based on the service's WSDL.

The NLP functions provided by HoG that are available in the Language Grid are JTok, ChaSen, TnT, TreeTagger, FreeLing, Chunkie, ChunkieRmrs, SProUT, LingPipe, Corcy, RASP, SDL, Sleepy, PET and RMRSmerge. These NLP functions can be accessed by adjusting the depth value of the Web service parameter. If the user of the Language Grid wants to use deeper NLP analyses, the depth value has to be set higher. Likewise, to access shallower NLP functions, the depth parameter can be decreased.

⁷ <http://www.w3.org/TR/xpath20/>

In the original HoG architecture, there is a multi session system and database that enables an offline processing mode. This means that the HoG engine does not have to use online NLP tools for analyzing sentences that had already been entered and analyzed before and were stored in an annotation database (caching via XML database). However, since there is no concept of multi-session and annotation database in grid computing, we have to disable these features in the current HoG service. HoG service in the Language Grid will have only single sessions and will not store any analyzed result in a database.

3.2. Use Scenario

To see how NLP services of HoG works in the Language Grid, we are experimenting with real case in the Language Grid. We create a scenario that uses the NLP tools provided by the HoG service in the multicultural activities, Aichi International Association⁸ and Kawasaki City⁹. These activities include creating and publishing their own language resource. In this experiment, we design a composition of language service by using annotation of HoG service in the Language Grid.

```
-<rmrs-list creator="PET" readings="1"> -<rmrs
cfrom="-1" confidence="0" cto="-1" reading="0"
surface=" ">
<label vid="1"/>
-<ep cfrom="0" cto="20" surface="Gaidansu wa 10 nich,
uten no baai mo okonaweremasu">
<gpred>proposition_m_rel</gpred>
<label vid="1"/> <var sort="e" tense="present"
vid="2"/> </ep>
-<ep cfrom="0" cto="4" surface="Gaidansu">
<realpred lemma="gaidansu" pos="n" sense="1"/>
<label vid="5"/> <var sort="x" vid="6"/> </ep>
-<ep cfrom="0" cto="4" surface="Gaidansu">
<gpred>udef_rel</gpred>
<label vid="8"/> <var sort="x" vid="6"/> </ep>
-<ep cfrom="5" cto="5" surface="wa">
<realpred lemma="wa" pos="p"/>
<label vid="12"/> <var sort="e" tense="u"
vid="13"/></ep>
-<ep cfrom="6" cto="6" surface="1">
<gpred>card_rel</gpred>
<label vid="15"/> <var sort="u" vid="18"/> </ep>
-<ep cfrom="6" cto="6" surface="1">
<gpred>plus</gpred>
<label vid="10001"/> <var sort="u" vid="21"/> </ep>
-<ep cfrom="7" cto="7" surface="0">
<gpred>card_rel</gpred>
<label vid="20"/> <var sort="u" vid="22"/> </ep>
-<ep cfrom="8" cto="8" surface="nichi"> . . .
```

Figure 3. Annotation result: a deep RMRS for Japanese

The first parallel text, let us say parallel text A, provides a comprehensive parallel text for disaster management system, while the second parallel text,

parallel text B, has a collective parallel text for education system. There is a need to improve the parallel text A by integrating this language resource with parallel text B. However, integrating these two language resources is not easy. These parallel texts have different scheme and categories. Therefore, to integrate these two language resources, we need to associate each parts of parallel text to an appropriate categories defined by another resources. By utilizing annotation tool of HoG service in the Language Grid, we can associate parts by parts in two parallel texts which might have different categories.

For example, in parallel text A, there is a sentence “Gaidansu wa 10th, uten no baai mo okonaweremasu (Guidance is on 10th, even though it rains)” located in the category of “gaidansu (guidance)”. On the other hand, this sentence belongs to the category of “undoukai (sports festival)” in Kawasaki City. By using a NLP tools in HoG service, we can annotate this sentence regardless what kind of categories they belong to. If we send this sentence to HoG service in the Language Grid and choose RMRSMerge as its NLP component, it will return the annotation. The result is described in Figure 3 (we omit the rest of the XML because it is too long to put in this paper).

```
-<rmrs-list creator="PET" readings="12"> -<rmrs
cfrom="-1" confidence="0.02242" cto="-1" reading="0"
surface=" ">
<label vid="1"/>
-<ep cfrom="0" cto="23" surface="Guidance is on 10th,
even though it rains">
<gpred>prop-or-ques_m_rel</gpred>
<label vid="3"/>
<var sort="e" tense="present" vid="2"/> </ep>
-<ep cfrom="0" cto="7" surface="Guidance">
<gpred>udef_q_rel</gpred>
<label vid="7"/> <var gender="n" num="sg" pers="3"
sort="x" vid="9"/> </ep>
-<ep cfrom="0" cto="7" surface="Guidance">
<realpred lemma="guidance" pos="n" sense="1"/>
<label vid="11"/> <var gender="n" num="sg" pers="3"
sort="x" vid="9"/> </ep>
-<ep cfrom="12" cto="13" surface="on">
<realpred lemma="on" pos="p"/>
<label vid="12"/> <var sort="e" tense="present"
vid="2"/> </ep>
-<ep cfrom="15" cto="16" surface="10"> . . .
```

Figure 4. Annotation result: a deep RMRS for English

This annotation result will then be used to compare with the annotation result for the same sentence in the parallel text in a different language. For example, when we submit the counterpart sentence of “Gaidansu wa 10 nich, uten no baai mo okonaweremasu” which is “Guidance is on 10th, even though it rains” in English, then the annotation result will be in Figure 4 (we omit also the rest). Consequently, these two results can be further compared to associate each word or phrase. The rest of the process is up to the application for integrating these annotation results into parallel text B, which is not

⁸ <http://www2.aia.pref.aichi.jp/tope/indexe.html>

⁹ <http://langrid.org/playground/kawasaki-parallel.html>

covered in this paper. This example scenario can be seen in the Figure 5.

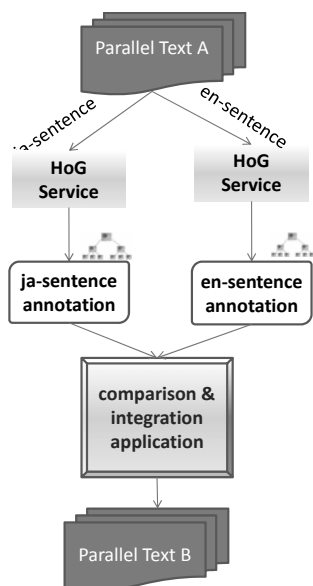


Figure 5. Integrated system for use scenario

4. Discussion

In the previous section, an example of using a particular component in HoG service is presented. We can extend this experiment to any case that needs NLP tools provided by the HoG service. We can use other NLP components of the HoG service in the Language Grid by adjusting the *depth* service parameter. By defining the depth parameter, we can also utilize multiple NLP components in HoG sequentially, since HoG applies the default strategy calling the NLP components from the shallowest components up to the requested depth.

In this experiment, it could be shown that having a composite service from HoG service and multiple services available in the Language Grid, i.e. parallel text services, is possible. It will be more interesting in the future when we have more researches in utilizing other services in the Language Grid working in harmony with the HoG service.

5. Conclusion

In this paper, we have proposed an integration of two frameworks, Heart of Gold and the Language Grid. We have examined several design alternatives for the integration and chose the most promising one.

The major contributions of this paper are as follows:

- To facilitate the use of NLP functions in HoG, we wrap HoG into a Web service in the Language Grid environment. By having a HoG service in the

Language Grid, we can exploit distributed language resources in NLP-based applications. Moreover, there are more options for morphological analyses in the Language Grid based on NLP tools in HoG service.

- We extend the standard interface of the Language Grid to accept an XML string as a result of an analysis by HoG. Because we can handle the result using XPath in a composite service, the result can easily be adapted to other services.

The next step of this work will be advancing the interface of the HoG service in the Language Grid in order to make HoG service a new standard in the Language Grid. Currently, the interface of HoG is adapting the interface of current service in the Language Grid, although the service that is offered by HoG is somewhat different. Another task that can be done on the basis of this prototype is to build applications for visualizing computed annotation results. Currently, the return value of HoG service is an XML string, 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 by HoG.

Lastly, we hope that by having experience in combining two frameworks, HoG and the Language Grid, we will gain confidence on how to redesign the Language Grid based on the Unstructured Information Management Architecture (UIMA) and redesign HoG as well by possibly also redesign HoG as well by adapting it to UIMA [5]. In UIMA [12], each element of an unstructured document will be associated with semantic results of analysis. This paradigm can be adapted to the Language Grid. Any words in source text translated by the Language Grid can be initially assigned a semantic value from UIMA. For simple example, the word “car” in text document can be associated with two analysis engines, the morphological analysis and translation engines. So, the result is the word “car” with semantic value “noun:en” and “kuruma: en → ja”. Having two frameworks, HoG and UIMA, in the Language Grid will increase the number of linguistic resources on the Web and increase the robustness of language services.

Acknowledgments

This research was partially supported by International Collaborative Research Grants from National Institute of Information and Communication Technology (NICT), and also from Global COE Program on Informatics Education and Research Center for Knowledge-Circulating Society. We are grateful to Wolfgang Wahlster for his initiative. We also thank Masaki Goto for his presentation of Aichi International Association and Kawasaki City.

References

- [1] K. Choukri, "European Language Resources Association History and Recent Developments", SCALLA Working Conference, KC 14/20, 2004.
- [2] T. Ishida, "Language Grid: An Infrastructure for Intercultural Collaboration", IEEE/IPSJ Symposium on Applications and the Internet (SAINT-06), 2006, pp. 96-100.
- [3] U. Schäfer, "Middleware for Creating and Combining Multi-Dimensional NLP Markup", Proceedings of the EACL-2006 Workshop on Multi-dimensional Markup in Natural Language Processing, Trento, Italy, 2006, pp. 81-84.
- [4] U. Schäfer, "Integrating Deep and Shallow Natural Language Processing Components - Representations and Hybrid Architectures", Doctoral Dissertation, Faculty of Mathematics and Computer Science, Saarland University, Saarbrücken, Germany, 2007.
- [5] U. Schäfer, "Shallow, Deep and Hybrid Processing with UIMA and Heart of Gold", LREC-2008 Workshop Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP, Marrakesh, Morocco, 2008, pp. 43-50.
- [6] "Business Process Execution Language for Web Services (BPEL), version 1.1.", <http://www.ibm.com/developerworks/library/ws-bpel/>, 2003.
- [7] Y. Hayashi, "Conceptual Framework of an Upper Ontology for Describing Linguistic Services", the First International Workshop on Intercultural Collaboration, 2007, pp. 246-260.
- [8] S. Sakai, M. Gotou, M. Tanaka, R. Inaba, Y. Murakami, T. Yoshino, Y. Hayashi, Y. Kitamura, Y. Mori, T. Takasaki, Y. Naya, A. Shigeno, S. Matsubara and T. Ishida, "Language Grid Association: Action Research on Supporting the Multicultural Society", Proceedings of the 2008 International Conference on Informatics Education and Research for Knowledge-Circulating Society (ICKS), 2008, pp. 55-60.
- [9] R. Khalaf, N. Mukhi and S. Weerawarana, "Service-Oriented Composition in BPEL4WS", Proceedings of the 12th World Wide Web Conference (WWW), Budapest, Hungary, 2003.
- [10] S. Shimohata, M. Kitamura, T. Sukehiro and T. Murata, "Collaborative Translation Environment on the Web", Proceedings of the MT Summit VIII, 2001, pp. 331-334.
- [11] H. Wang, J. Z. Huang, Y. Qu and J. Xie, "Web Services: Problems and Future Directions", *Journal of Web Semantics*, 2004, pp. 309-320.
- [12] D. Ferrucci and A. Lally, "Building an Example Application with the Unstructured Information Management Architecture", *IBM Systems Journal* 43(3), 2004, pp. 455-475.
- [13] A. Copestake, *Report on the Design of RMRS. Technical Report D1.1b*, University of Cambridge, Cambridge, UK, 2003.
- [14] C. Pollard and I. Sag, *Head-Driven Phrase Structure Grammar*, University of Chicago Press, 1994.
- [15] T. Ishida, A. Nadamoto, Y. Murakami, R. Inaba, T. Shigenobu, S. Matsubara, H. Hattori, Y. Kubota, T. Nakaguchi, and E. Tsunokawa. "A Non-Profit Operation Model for the Language Grid", the First International Conference on Global Interoperability for Language Resources (ICGL-08), 2008, pp. 114-121.