

## Towards Service Supervision for Public Web Services

Masahiro Tanaka<sup>1</sup>, Yohei Murakami<sup>2</sup> and Toru Ishida<sup>1,2</sup>

<sup>1</sup>*Department of Social Informatics, Kyoto University,  
Kyoto 606-8501 Japan*

<sup>2</sup>*Language Grid Project, National Institute of Information and Communications Technology,  
3-5 Hikaridai, Seika-cho, Kyoto, Japan*

*mtanaka@ai.soc.i.kyoto-u.ac.jp, yohei@nict.go.jp, ishida@i.kyoto-u.ac.jp*

### Abstract

*Public Web services are not designed to be used with specific other Web services in a composite Web service. This leads to the following requirements for the proper control of a composite Web service; 1) adaptation to changes in Web services, 2) coordinating the contexts of internal processing, 3) flexible execution of human tasks. These controls must be applied following the policies of stakeholders, such as service providers. Some previous works proposed a method that adds processes to a composite Web service. Unfortunately, they fail to implement the controls needed and do not consider the policies of stakeholders. In this paper, we propose a meta-level architecture named Service Supervision, which controls a composite Web service based on the policies of stakeholders in order to achieve the requirements. To show the effectiveness of our architecture, we mention applications that currently apply Service Supervision to composite Web services consisting of machine translator Web services and human tasks.*

### 1. Introduction

Composite Web services have been mainly established within companies or among companies that have common goals. Recently, however, infrastructures for public Web services that are intended to be accessed by unspecified users have been developed. Web services which wrap programs or contents inside certain organizations have been realized on such infrastructures.

For example, various organizations now provide different linguistic services such as dictionaries or machine translators for Language Grid[5], an infrastructure for Web services. Composite Web services are also provided, such as a translation service for a special domain that combines a machine translator with a technical term dictionary.

Public Web services are not designed to be used with specific other Web services in a composite Web service. This leads to the following requirements for the runtime control of composite Web services composed of public Web services.

**Adaptation to changes in Web services** The availability and behavior of a public Web service can be changed by its provider or operator without the agreement of the users. A composite Web service which consists of public Web services has to adapt to the changes that may occur in its constituents.

**Coordinating contexts of internal processing** Some Web services assume the input specifies the context needed for internal processing. Context coordination is essential in a composite Web service because its constituent public Web services may assume different contexts.

**Flexible execution of human task** Composing public Web services often requires interpretation or evaluation by humans. However, in case of human tasks, the schedule and the assignment should be flexibly managed because available time of humans and quality of the results are difficult to predict.

Moreover, these controls must be applied following the policies of the stakeholders, such as service providers, composite service designers and operators.

Some previous works such as AO4BPEL[4] and Dynamo[2] adopted Aspect-oriented Programming (AOP) in order to add some processes to a composite Web service. But inserting a process at a specific point is not enough to satisfy the above requirements.

In this paper, we propose a meta-level architecture for controlling composite Web services. Named Service Supervision, the architecture realizes the controls needed by replacing service invocation with one of several control functions. The control functions are applied based on the policies of the stakeholders.

The rest of this paper is organized as follows. Section 2 introduces the Language Grid as a good example of the need for Service Supervision in composing public Web services. Next we explain the requirements for organizing public Web services and Service Supervision which achieves the requirements in Section 3 and 4, respectively. Section 5 introduces two applications of Service Supervision. In Section 6, we discuss previous research related to our work. Finally we conclude this paper in Section 7.

## 2. Case study: The Language Grid

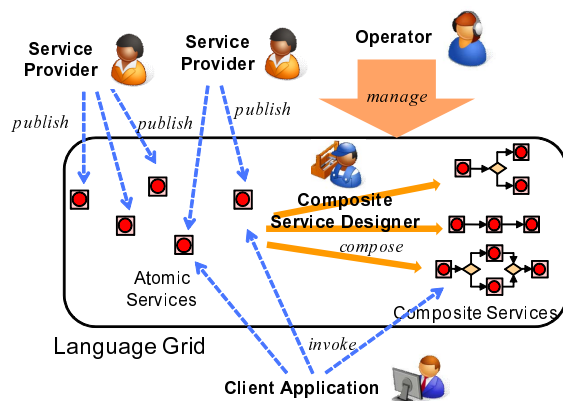
The Language Grid[5] is an infrastructure for public Web services. Various language resources, such as dictionaries and machine translators, are wrapped and made available as Web services on the Language Grid. Composite Web services which combine several Web services are also available. Language resources and Web services on the Language Grid are managed with a Web application named the Language Grid Service Manager (Fig. 1)<sup>1</sup>.

Resource Name	Resource Type	Languages (in Language Code)	Provider	Status
<a href="#">Dialog Corpus for Medical Scenes</a>	DIALOG CORPUS	(ja), (zh), (en), (ko), (pt)	<a href="#">Kyoto Center for Multicult...</a>	Run
<a href="#">EDR Japanese/English Word Dictionary</a>	BILINGUAL DICTIONARY	(en<->ja)	<a href="#">Computational Linguistics ...</a>	Run
<a href="#">ICTCLAS</a>	MORPHOLOGICAL ANALYZER	(zh)	<a href="#">NLP Group, Institute of Co...</a>	Run
<a href="#">J-Server</a>	TRANSLATOR	(ja<->en), (ja<->ko), (ja<->zh-CN)	<a href="#">Ishida and Matsubara Labor...</a>	Run
<a href="#">J-Server</a>	TRANSLATOR	(ja<->en), (ja<->ko), (ja<->zh-CN)	<a href="#">Computational Linguistics ...</a>	Run
<a href="#">KLT version 2.1</a>	MORPHOLOGICAL ANALYZER	(ko)	<a href="#">Seung-Shik Kang Laboratory ...</a>	Run
<a href="#">Kawasaki City CEC Elementary S Education Series: Social Studies</a>	PARALLEL TEXT	(ja<->zh), (ja<->ko), (ja<->es), (zh<->ko), (zh<->es), (ko<->es)	<a href="#">Kawasaki City Comprehensiv...</a>	Run

**Figure 1. Language resources registered to Language Grid Service Manager**

Figure 2 shows stakeholders of the Language Grid. Service providers wrap their language resources and publish them as Web services on the Language Grid. Composite service designers provide composite Web services which consist of the Web services provided by service providers or other composite Web service designers. Web services available on the Language Grid are invoked by client applications. The service providers and the composite Web service designers manage their own Web services. On the other hand, the operator of the Language Grid manages all Web services on the Language Grid.

Each stakeholder has policies for the management of Web services, such as limitations about usage. The Web



**Figure 2. Stakeholders of Language Grid**

services should be executed following the policies of the related stakeholders.

Web services on the Language Grid are not specialized in a certain combination with other Web services. As the result of this, a composite Web service consisting of public Web services sometimes has problems at runtime.

Take the example of a composite Web service in which the result of a machine translator Web service is given to another machine translator Web service. This service is called a multi-hop translation service and used for translation between languages unsupported by any single machine translator Web service.

This composite Web service fails if either of the translator services is unavailable at runtime. Unfortunately, public Web services offer no guarantee of availability to the users.

Moreover, the second machine translator Web service may output an incorrect translation if output of the first one includes ambiguous words. Machine translator Web services usually translate the sentences input following the guessed contexts because contexts are too complicated for users to explicitly specify. Furthermore, they do not output the context assumed either because they are not designed to be used with other machine translator Web services. As the result of this, translators developed by different organizations will make different guesses.

Composite Web services which use machine Web translator services often require humans to evaluate and/or correct the results of the translation. BPEL4People[1], an extension of WS-BPEL, allows such human interaction to be defined in the same way as a Web service. But the available time and the ability to evaluate or correct the translation result differ greatly in individuals. Therefore controls tailored to human behavior are required.

The more public Web services are available, the more serious the problems described above get because of difficulty in managing and maintaining the public Web services.

<sup>1</sup><http://langrid.org/operation/service.manager/>

### 3. Requirements for organizing public Web service

In this section, we explain the key requirements for organizing public Web services in a composite Web service.

#### 3.1. Adaptations to changes in Web services

Public Web services are managed by the service provider and the operator, not the users. Therefore the following adaptations to unexpected changes in public Web services are required.

**Dynamic selection of public Web services** On an infrastructure for public Web services, like the Language Grid, new public Web services are registered daily and some will become unavailable. Therefore, we should check a list of available Web services at runtime and select those public Web services that are available and offer the functions desired.

#### Absorbing changes in the behaviors of Web services

Service providers sometimes update the program or the contents of their Web services. In order to use the service in the same way as previous, we have to absorb the changes in the behavior of the Web service.

#### 3.2. Coordinating contexts of internal processing

Some Web services assume the input's context for internal processing. The contexts may not be explicitly specified by parameters. For example, a machine translator Web service assumes the context based on the sentence input. The context is used to decide the interpretation of ambiguous words.

Different public Web services will decide the context in different ways and cannot pass the context to other Web services because public Web services are not designed to be used with specific other public Web services. Therefore, we have to check if the public Web services follow a common context and correct the results of the Web services if needed.

#### 3.3. Flexible execution of human task

Making human interaction a part of a composite Web service is useful because humans are good at evaluating and/or correcting the results of Web services. But available time of humans and quality of human tasks are difficult to predict. Therefore we need allow human tasks in a composite Web service to be executed more flexibly than Web services.

**Re-executing tasks** Person A, who is to execute human task  $HT_A$ , can have multiple tasks in his task box.

Suppose another person B is to execute another human task  $HT_B$  which is dependent on the result of  $HT_A$ . In this case, A may try to finish his tasks as soon as possible in order to allow B to start working. If A finishes all of his tasks and B is not available (e.g. not logged on), the composite Web service containing  $HT_A$  and  $HT_B$  should allow A to spend more time on re-executing the tasks which A once finished in order to improve the quality of the result. This makes it possible to utilize unoccupied hours of A.

**Runtime assignment of tasks** A human task defined in BPEL4People requires the assignment of a particular person or group of people. The quality of the result depends on the person/group specified so we should assign the best person/group to each task. Work quality depends, in large measure, on the person's workload, so person/group selection should be made at runtime to avoid overloading.

### 4. Service Supervision

In this section, we propose a meta-level architecture, named Service Supervision, for controlling composite Web services.

As we explained in the case of the Language Grid in Section 2, public Web services have various stakeholders. The requirements described in Section 3 must be achieved following the policies of the stakeholders. Therefore our architecture applies control functions to a composite Web service at runtime according to the policies of stakeholders. The control functions are realized at a meta-level without modifying the composite Web service.

The overview of the architecture is shown in Fig. 3. The architecture has four layers. Each layer consists of one or more Web services as described in Table 1.

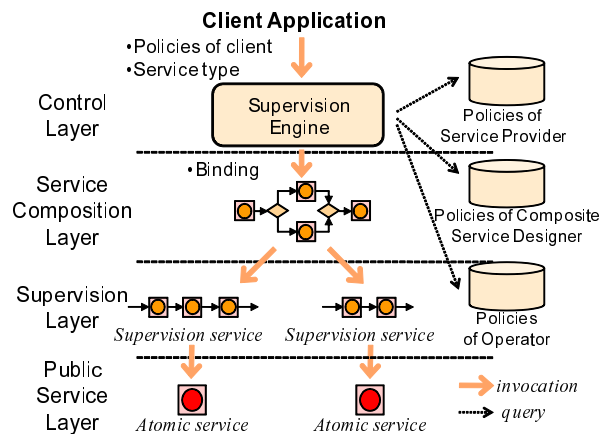


Figure 3. Layers for Service Supervision

**Table 1. Web services in layers for Service Supervision**

Layer	Web services in the layer
Control Layer	The Web service named Supervision Engine receives a request from a client application and invokes a composite Web service in Service Composition Layer based on the service type desired. According to the policies of stakeholders, the Supervision Engine sets endpoints for Web services contained in the composite Web service in Service Composition Layer to composite Web services in Supervision Layer using dynamic binding. Dynamic binding makes it possible to set endpoints of Web services in a composite Web service before invocation and is implemented on the Language Grid.
Service Composition Layer	Composite Web services which consist of Web services provided by service providers or composite Web service designers. They invoke Web services in Supervision Layer and Public Service Layer according to policies.
Supervision Layer	Composite Web services named supervision services which realize various controls. They invoke Web services in Public Service Layer or another supervision service for additional control specified by policies.
Public Service Layer	Web services provided by service providers or composite service designers.

A policy is defined as a set of rules. A rule has a condition and dynamic binding which is applied when the condition is satisfied. Limitations of users, types of services, and status of services are specified as the conditions.

Policies are classified into four levels which correspond to service providers, operators, composite service designers and client applications. Policies are prioritized in this order. Service providers and operators describe policies on access control or load balancing for the Web services they manage. Composite service designers describe control policies for specific tasks. Client applications set policies on their use in composite Web services. When two dynamic binding settings described in policies conflict, the one with the highest priority is applied.

Control functions that achieve the requirements described in Section 3 are implemented as supervision services. Table 2 shows the correspondence between the control functions and the requirements satisfied.

## 5. Two current applications

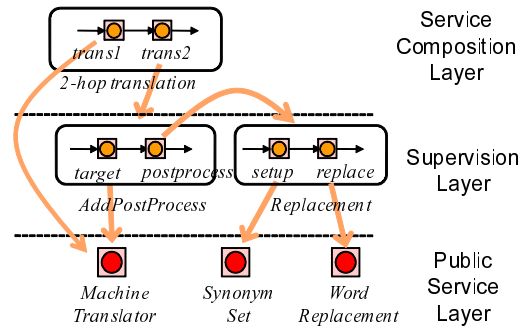
This section describes the control of two composite Web services as applications of Service Supervision.

### 5.1. Context coordination for multi-hop translation

Consider the 2-hop translation service that uses two machine translator Web services and translates language A into language C, via language B. When we have a set of tuples of synonyms in languages A, B and C, we can correct translation errors created by ambiguous words. If word  $w_A$  in the original sentence in language A is translated into

$w_B, w_C$  in languages B and C, respectively, and the tuple  $(w_A, w_B, w_C)$  is not found in the synonym set, the output of  $w_C$  by the second translator Web service is wrong. This is because  $w_A$  and  $w_B$  represent the context of translation. Therefore, we can coordinate the context of the second translator with that of the first one by replacing  $w_C$  with  $w'_C$  based on the tuple  $(w_A, w_B, w'_C)$  in the synonym set.

The solution is to realize overall effective control by Service Supervision. Invocations between Web services in Service Composition Layer and lower layers are shown in Fig. 4. The Web services in the layers are described in Table 3.



**Figure 4. Service supervision for multi-hop translation**

### 5.2. Protocol control for collaborative translation

Two people A and B, who understand only language  $L_A$  and  $L_B$  respectively, can work together in order to translate

**Table 2. Control functions and requirements satisfied**

Control functions	Requirements achieved
Changing invocation settings	Invocation settings such as endpoint of a Web service or human assigned to a task can be changed in a supervision service. This realizes dynamic selection of Web services based on availability information retrieved from the service repository, and task assignment to humans based on profiles or status of the people/groups.
Adding pre/post process	In a supervision service, any process is added before and after invocation of a Web service in a composite Web service in Service Composition Layer. This makes it possible to adapt to changes in the behavior of a Web service and coordinate context of internal processing by monitoring and modifying parameters and results.
Spooling results of human tasks	A supervision service which has a loop and monitoring function of humans performing tasks allows the result of human tasks to be sent to the following Web service or human tasks only after the human performing the following task logs on.

**Table 3. Web services for controlling multi-hop translation service**

Layer	Web services in the layers
Control Layer	The Supervision Engine invokes 2-hop translation service and sets the endpoint for the second machine translator Web service to a supervision service for correction of words in the result of translation using a synonym set.
Service Composition Layer	A composite Web service for 2-hop translation service is invoked by the Supervision Engine. The endpoint of the second one is set to a supervision service for adding a post process.
Supervision Layer	Two supervision services are used. The first one is for adding any post process to a service invocation. This service invokes a machine translator and another supervision service. The second one is for replacement according to criteria given by other service. This invokes a synonym set Web service to find words to be replaced and a term replacement Web service to rewrite a sentence by replacing words.
Public Service Layer	This layer contains a machine translator Web services and Web services for retrieving synonyms and replacing words.

a document in  $L_A$  into  $L_B$ . First A translates the document in  $L_A$  into  $L_B$  using a machine translator and sends the result to B. Next B checks the fluency of the translated document, refines it if it is not fluent enough, and sends it to A after translating it into  $L_A$  using a machine translator. Then A checks the accuracy of the document from B by comparing it against the original document. A refines the document if it is not accurate enough. By iterating this process, they can improve the quality of the translation. We refer to this process as collaborative translation.

The process of collaborative translation can be represented in BPEL using BPEL4People. Refinement by humans is defined as a human task.

Assume multiple documents are to be concurrently translated by A and B following the protocol of collaborative translation. When A have multiple refinement tasks in his task box, A may try to finish the tasks as soon as possible to allow B to start working. But if B is not logged on after A finishes all tasks, A should be allowed to spend more time on refinement which he once finished. Table 4 describes Web services in the layers for Service Supervision.

## 6. Related works

Several papers have described the addition of processes to a composite Web service at a meta-level without modifying the composite Web service. AO4BPEL[4] introduced Aspect-Oriented Programming (AOP) into BPEL. It allows the composite service designer to insert any process before/after an activity specified as a pointcut. Dynamo[2] is also based on the concept of AOP. It monitors exchanges of messages between a BPEL process and external Web services and checks if the messages satisfy constraints.

Both works described above focus on “weaving” some processes into a composite Web service. On the other hand, the goal of our work is to control a composite Web service consisting of public Web services based on the policies of stakeholders. For this purpose, weaving processes into specific points is not enough. Moreover, we need to allow stakeholders to describe their control policies.

Program Supervision[6] tries to compose legacy program components, not Web services. It automatically finds a plan that can appropriately process the given data by com-

**Table 4. Web services for controlling protocol for collaborative translation**

Layer	Web services in the layers
Control Layer	The Supervision Engine invokes a composite Web service for collaborative translation and sets the endpoint for human tasks to a supervision service for spooling results.
Service Composition Layer	A composite Web service for collaborative translation which invokes a machine translator Web service and two human tasks.
Supervision Layer	A supervision service which continues to put the task into the task box of person A responsible for a human task iteratively until person B responsible for the following human task logs on. After B logs on, this supervision service deletes the task and goes to the next step.
Public Service Layer	A machine translator Web service and human tasks for refinement performed by two people.

binning program components. However, it did not consider some characteristics which public Web services can have, such as dynamic changes.

Adaptive Workflow[7] focuses on workflows which are mainly executed by humans. It aims to adapt unexpected exceptions and changes of the environment. Some previous works adopt case-based reasoning, rule-based system and planning to realize the adaptations[3]. But they considered neither policies of stakeholders nor interaction between Web services and human tasks.

## 7. Conclusion

Public Web services are not designed to be used with specific other Web services in a composite Web service. This leads requirements for runtime control of composite Web services. In this paper, we proposed a meta-level architecture named Service Supervision, which satisfies the requirements of the control.

The major contributions of this work are as follows:

- In order to adapt changes in Web services, we realized dynamic selection of Web services and assignment of human tasks by changing invocation setting at runtime.
- We made it possible to add pre/post processes to Web service invocation to absorb changes in the behavior of Web services and coordinate the contexts of Web services.
- We allowed a person responsible for a human task to re-execute the task to utilize unoccupied hours of the person. This is realized by spooling the result until other person responsible for the following task logs on.

The architecture we proposed ensures that a composite Web service can be controlled to satisfy the policies of stakeholders. Various stakeholders, such as service providers, composite service designers and operators are involved in providing public Web services. Therefore, the above control of a composite Web service is applied following the policies of the stakeholders.

We also showed two applications of Service Supervision to control composite Web services consisting of machine translator Web services and human tasks.

In future work, we will tackle the coordination of the policies of various stakeholders.

## Acknowledgments

This work was supported by Grant-in-Aid for JSPS Fellows and Global COE Program “Informatics Education and Research Center for Knowledge-Circulating Society”.

## References

- [1] WS-BPEL extension for people (bpel4people), version 1.0. <http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/>, 2007.
- [2] L. Baresi, S. Guinea, and P. Plebani. Policies and aspects for the supervision of BPEL processes. In *the 19th International Conference on Advanced Information Systems Engineering (CaiSE07)*, pages 340–354, 2007.
- [3] F. Casati, S. Ilnicki, L. J. Jin, and V. Krishnamoorthy. Adaptive and dynamic service composition in eFlow. In *12th International Conference on Advanced Information Systems Engineering (CaiSE02)*, pages 13–31, 2000.
- [4] A. Charfi and M. Mezini. AO4BPEL: An aspect-oriented extension to bpel. *World Wide Web*, 10(3):309–344, 2007.
- [5] T. Ishida. Language Grid: An infrastructure for intercultural collaboration. In *IEEE/IPSJ Symposium on Applications and the Internet (SAINT-06)*, pages 96–100, 2006.
- [6] M. Thonnat, V. Clement, and J. v. d. Elst. Supervision of perception tasks for autonomous systems: The OCAPI approach. In *the 3rd Annual Conference of AI, Simulation, and Planning in High Autonomy Systems*, pages 210–217, 1992.
- [7] W. M. P. van der Aalst, T. Basten, H. M. W. Verbeek, P. A. C. Verkoulen, and M. Voorhoeve. Adaptive workflow. on the interplay between flexibility and support. *the first International Conference on Enterprise Information Systems*, 2:353–360, 1999.