2012 IEEE Ninth International Conference on Services Computing

Service Grid Federation Architecture for Heterogeneous Domains

Yohei Murakami, Masahiro Tanaka
Information Services Platform Laboratory
National Institute of Information and Communications Technology (NICT)
Kyoto, Japan
{*yohei, mtnk*}*@nict.go.jp*

Donghui Lin, Toru Ishida
Department of Social Informatics
Kyoto University
Kyoto, Japan
{*lindh, ishida*}*@i.kyoto-u.ac.jp*

***Abstract*—Service grid is an infrastructure for service-oriented collective intelligence. It provides a set of enabling functionalities to support coordination of services, such as service registries, service composition, access control, and monitoring. To form the service-oriented collective intelligence, various types of services need to be connected on the service grid, and managed by the service grid operator. However, it is difficult for single service grid operator to gather and organize services in various domains. Therefore, building service grids in different domains and connecting these service grids are essential for expanding service-oriented collective intelligence across domains. To this end, we have designed a service domain model to specialize general-purpose service grid to a specific domain and realize interoperability among service grids. Moreover, we have also developed service grid federation architecture to share service registries, compose services across service grids, and control and monitor accesses to the composite services. Finally, we have applied the proposed architecture to the language service domain to construct the Language Grid.**

***Keywords*-service grid; federation;**

I. Introduction

Collective intelligence creates new value by aggregating and assembling local knowledge from a lot of participants. Wikipedia, which is content-oriented collective intelligence aggregating texts, pictures, and movies, is being created by people all over the world. Meanwhile, service-oriented collective intelligence, which assembles Web services, has not been realized even though cloud computing provides us with a huge computation environment enough to aggregate services worldwide. To make a success of the service-oriented collective intelligence, we need a service middleware to aggregate and combine services in open environment.

Some previous works proposed middleware to compose services, such as ESB (Enterprise Service Bus). By plugging a service into the ESB, ESB allows the service to communicate with other services. However, since ESB is middleware to be used within enterprises, ESB operators have to integrate services into ESB's mediation after reaching agreement on policies with service providers. On the other hand, service-oriented collective intelligence is built in open environment. To accelerate the aggregation of services, the middleware has to enable service providers to register and provide their services voluntarily. Moreover, it must allow the service providers to set access control policy to prevent the abuse.

To satisfy those requirements, we developed service grid. Service grid provides enabling functionalities to realize service composition in open environment [1]. Service grid enables users to compose services to meet their requirements within constraints specified by the service providers. The key requirement in Service grid is a description of its interface and capability. This information is essential in order to compose a set of services to meet users' requirements.

However, there is no standard interface in open environment even though services provide the same functionalities. As a result, users have to implement composition logic for each service combination. Due to the lack of the reusability of composition logic, the collective intelligence is not expanded in direct proportion to the number of shared services. In order to solve this problem and accelerate forming collective intelligence, we extend service grid so that service grid operators can standardize interface for each service type and organize them. Moreover, we propose service grid federation consisting of distributed service grids operated by different operators because a single operator has limitation to standardize and organize various service domains. To this end, we address the following research issues.

Design a service domain model
To construct service grid for each service domain, it is necessary to make service grid middleware independent of the domain. By installing this domain definition of services and resources into the service grid middleware, service grid operators have to specialize general-purpose service grid to a certain domain.

Develop service grid federation architecture
We need federation architecture to coordinate service grids. Since each service grid is operated by a different operator, the federation architecture clarifies which information should be shared, and which function should be performed by which service grid.

The rest of this paper is organized as follows. Section II introduces service oriented collective intelligence and ser-

978-0-7695-4753-4/12 $26.00 © 2012 IEEE
DOI 10.1109/SCC.2012.101

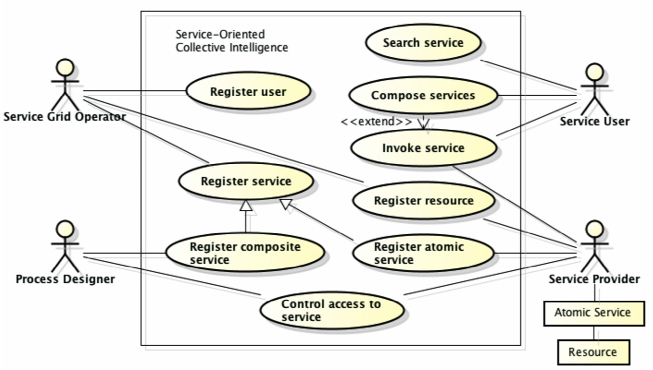

Figure 1. Use Cases of Service-Oriented Collective Intelligence

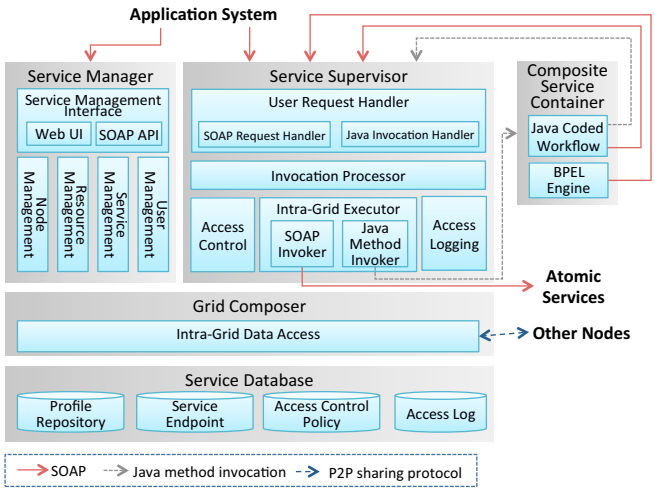

Figure 2. Service Grid Architecture

vice grid to realize it. Section III presents a service domain model to specialize general-purpose service grid to a specific domain. Section IV explains service grid federation architecture to share information and compose services across service grids. Section V illustrates how to form collective intelligence with the proposed architecture by using a case study of the Language Grid. After introducing related works in Section VI, we conclude this paper and point to some further work.

II. Service-Oriented Collective Intelligence

Service-oriented collective intelligence is realized by aggregating and assembling various types of services created by a lot of service providers. Key properties of collective intelligence are independence, decentralization, diversity, and aggregation [2]. Realizing diversity and aggregation of services is very significant because the first two have been satisfied with characteristics of services.

To form service-oriented collective intelligence, an operator needs to operate a platform to aggregate and compose services. The platform has to allow service providers to register their services voluntarily for sharing various services. Similarly, it also has to permit process designers, which is also service providers, to register their workflows to compose services. Note that the operator needs to acknowledge their registration in order to maintain the quality and security of services. Moreover, the platform has to provide service providers with access control so that they can prevent the abuse. On the other hand, service users are allowed to compose services within constraints specified by service providers. Fig. 1 shows these use cases.

A. Service Grid

To form service-oriented collective intelligence, we developed the service grid in previous work [1]. Fig. 2 shows the service grid architecture consisting of five parts: Service Manager, Service Supervisor, Grid Composer, Service Database, and Composite Service Container.

The Service Manager manages node, user, service and resource information registered in the service grid. The service information includes access control settings and access logs. Since the information is registered through the Service Manager, it plays a front-end role of any functions other than service invocation. The Service Supervisor controls service invocations according to the access control settings registered by the service providers. Before invoking the services on the Composite Service Container, it validates whether the request satisfies providers' access control settings. The Grid Composer connects the nodes within the service grid to realize P2P grid network. The Service Database is a repository to store various types of information registered through the Service Manager and service invocation logs. The Composite Service Container provides composite service deployment, composite service execution, and dynamic service binding so that service users can customize services.

In the remaining parts of this section, we provide the details of the Service Manager, Service Supervisor, Grid Composer, and Composite Service Container.

1) Service Manager: The Service Manager consists of components managing various types of information necessary for the service grid, such as node, resource, service, and user information. The Node Management handles node information of the service grid. Based on this information, the Grid Composer decides whether to save information received from other nodes, and whether to distribute information to other nodes. The Resource Management and Service Management handle resource and service information registered to the service grid and the connected service grid. The information includes access control settings, service endpoints, intellectual properties associated with the resources, and access logs. Based on this information, the Service Supervisor validates service invocation, locates service

endpoints, and attaches intellectual property information to service responses. Finally, the User Management manages user information registered to the service grid. Based on this information, the Service Supervisor authenticates users' service requests.

2) Service Supervisor: The Service Supervisor controls service invocation. The control includes user authentication, access control, endpoint locating, load balancing, and access logging. The User Request Handler receives service requests from service users, and then authenticates the users. The requests are sent to the Invocation Processor. The Invocation Processor executes a sequence of pre-process, service invocation, post-process, and logging process. The access control is a pluggable component implemented as the pre-process, or the post-process. After passing the access control, the Intra-Grid Executor invokes the service within its service grid. To invoke the service, the Intra-Grid Executor locates the service endpoint. If there are multiple endpoints associated with the service, it chooses the endpoint with the lowest load. This load balancing is also a pluggable component. Currently, we provides two types of load balancing components; choosing the endpoint whose response time is fastest, and round robin. Finally, it invokes the service using Java Method Invoker implementation or SOAP Invoker implementation, which are selected according to the endpoint location.

3) Grid Composer: The Grid Composer creates a P2P grid network within the service grid. The Intra-Grid Data Access provides interfaces to read and write the Service Database in the service grid. In writing data, the Intra-Grid Data Access broadcasts the data to other nodes using the P2P network so that it can share the data with other nodes in the same service grid. It results in that service users can improve latency by sending their requests to a node located near the service. In this way, usage of the P2P network framework contributes to scalability of the service grid while keeping data consistency.

4) Composite Service Container: The Composite Service Container contains composite services whose abstract workflows are implemented by Java or WS-BPEL. The BPEL workflows are executed by BPEL Engine like active BPEL. In invoking a component service of a composite service, Java coded workflow or BPEL Engine can select a concrete service based on binding information included in a service request.

III. Service Domain Model

Workflows depending on each service interface slow down emergence of service-oriented collective intelligence. Especially, this problem becomes critical as the number of services increases. Therefore, we propose a service domain model for operators to define their target service domain. The service domain model is not just type system of data exchanged between services, but type system of service interfaces, service meta data, and resource meta data. Using

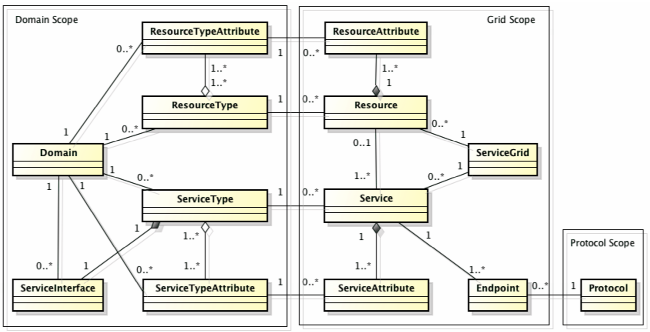

Figure 3. Relation between Domain Model and Instances

this model, the operators can classify services in the target domain by functionalities, and define standard interfaces and meta data for each service type. These standard service interfaces can increase reusability of workflows. It results in service users can compose services by searching services with the meta data and changing the services belonging to the same service type.

The service domain model for service grid is divided into service type definition and resource type definition. Moreover, services consist of atomic services that provide a method to access service providers' resources, and composite services which are implemented by a workflow to invoke several atomic services. On the other hand, resources contain data, software, and human resources accessed via atomic services. These service type and resource type are defined in the service domain model. The definition of resource type has more than one resource type attributes that indicate attributes of resources belonging to the resource type. Meanwhile, the definition of service type also has more than one service type attributes that indicate attributes of services belonging to the service type. Moreover, the service type definition has to contain single service interface which specifies how to invoke the service type. This interface is used to wrap service providers' resources as a web service and describe service composition workflows. This domain model can specialize general-purpose service grid to a certain domain. Fig. 3 shows whole service domain model.

Services and resources on service grid are necessarily related to service type and resource type defined in the service domain model. Resources can be related to several services to access the resources. This means a resource can provide several different types of services. On the other hand, service can be related to several endpoints compliant with service interface of the corresponding service type. This means that a service can have several endpoints for load balancing and provides several protocols.

In this way, services and resources on service grid have association with the domain model. Service grid uses this

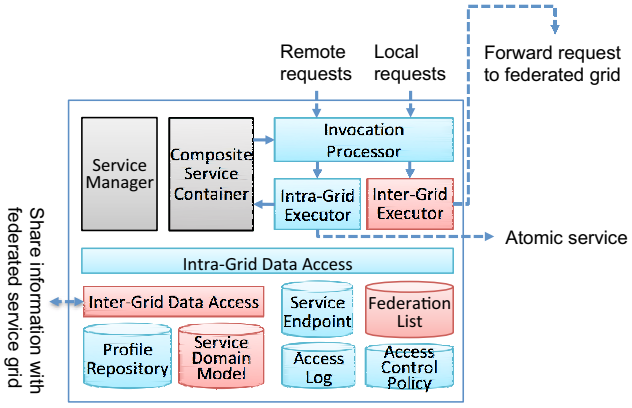

Figure 4. Service Grid Federation Architecture

association to change services belonging to the same service type, search and select services with a specific attribute of service type. These facilities can reduce the cost to compose services, and accelerate the augmentation of the collective intelligence.

IV. Service Grid Federation Architecture

The service domain model is defined in each domain. By coordinating various service grids, we can share services among multiple domains and accelerate formation of collective intelligence. To this end, we propose a new distributed service management architecture, called *Service Grid Federation* architecture. This architecture allows the transparent use of services through the federation when services in a single service grid are insufficient to meet its users' requirements, while providing admission control at each service grid in the federation.

Fig. 4 shows the service grid federation architecture extended from the service grid architecture indicated in Fig. 2. Inter-Grid Executor and Inter-Grid Data Access, indicated by red rectangles, are newly introduced into the existing service grid architecture. In addition, the introduction of service domain model and the federation puts repositories for service domain definition and a federated service grid list.

The Inter-Grid Executor is responsible for invoking services across service grids when the destination of the locally submitted requests is another federated service grid. This contrasts to the Intra-Grid Executor responsible for invoking services on the same service grid. *Local request* refers to the request submitted by the users of the same service grid, while *remote request* refers to the incoming request from another federated service grid.

The Inter-Grid Data Access is responsible for sharing information with the other federated service grids. The information is open to every stakeholder, such as user, resource, and service profile information, and service domain definition. This contrasts to Intra-Grid Data Access responsible for sharing information with other nodes in the same service grid. The information shared by Intra-Grid Data Access is private one for each stakeholder, such as service endpoints, access logs, and access control policies for service providers, usage histories for service users, and a federated service grid list for operators.

In this section, we present the details of information sharing and service composition in the federation.

A. Information Sharing

Once establishing federation between service grids, they issue certificates each other. These certificates are used to certify a member of the federation in sharing information and composing services across the service grids. Then, the service grid operator requesting federation (called *affiliated operator*) obtains service domain definition and profile information including users, resources, and services from the target service grid operator. The affiliated operator publishes the information to its service users (called *affiliated users*) so that the affiliated users can invoke services managed by the target service grid operator.

On the other hand, the target service grid operator obtains only user profile information managed by the affiliated operator, and publishes it to its service providers so that they can control access from the affiliated users. Such an asymmetric information sharing can prevent a service list the target operator is not interested in from being mixed into its service list. Therefore, bidirectional federation is necessary for both of the service users and the affiliated users to invoke services across the service grids.

After establishing the federation, information is shared via Inter-Grid Data Access based on the above information sharing policy. Meanwhile, private information, such as access control policy, access log, service endpoint information, user authentication information, and so on, is not shared among service grids.

B. Service Composition across Service Grids

Fig. 5 shows messaging between components when invoking services across service grids. In this figure, an application system of a service grid invokes a composite service on the affiliated service grid and atomic service on the service grid.

In composing services across service grids, Invocation Processor decides which Intra-Grid Executor or Inter-Grid Executor to use according to destination of a service request. If the destination is the same service grid, it selects the Intra-Grid Executor. Otherwise, it selects Inter-Grid Executor to forward the request to the federated service grid where the service is deployed.

Fig. 6 shows the details of processing flow among Invocation Processor, Intra-Grid Executor, and Inter-Grid Executor.

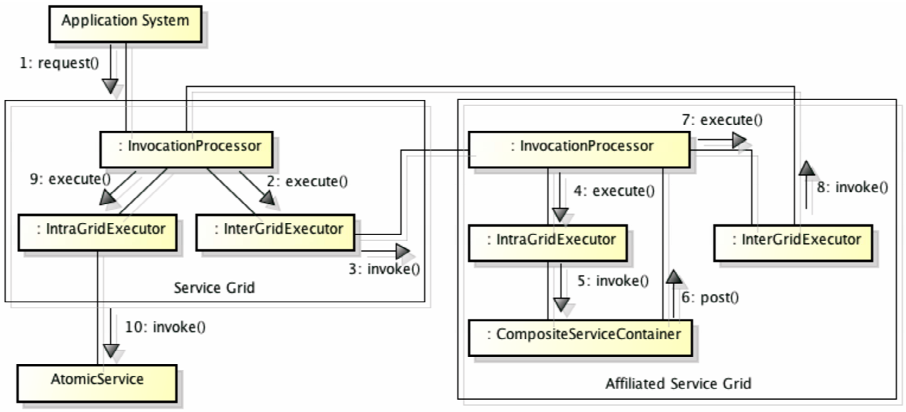

Figure 5. Service Composition in Service Grid Federation

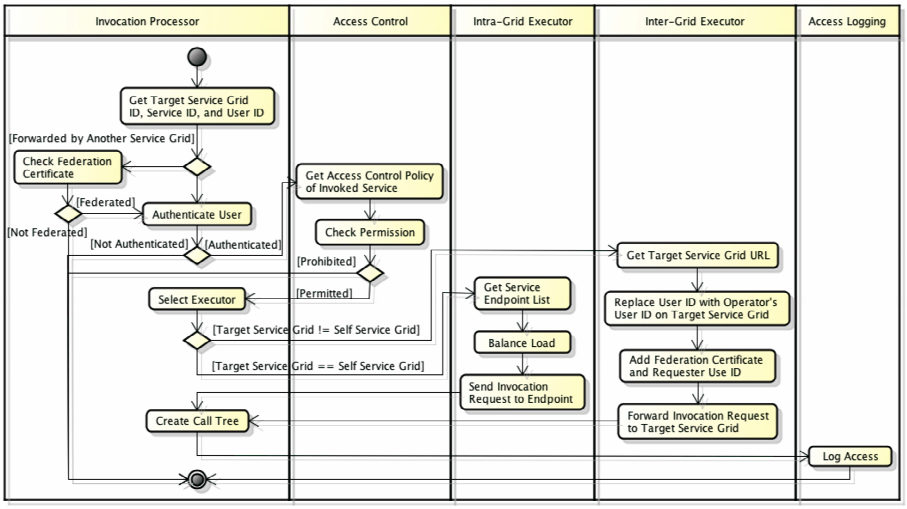

Figure 6. Service Composition Process in Service Supervisor

When Invocation Processor receives a service request from an application system, it checks whether it is a local request or a remote request, and authenticates the requester. After that, Invocation Processor performs access control, service execution, and access logging sequentially. Since access control policies are not shared in the federation, it skips the access control if the destination is another federated service grid. In executing the service, it selects Intra-Grid Executor if the destination is the same service grid. Otherwise, it selects inter-Grid Executor to forward the request to the different service grid.

Intra-Grid Executor obtains a list of endpoints related to the service to be invoked, and then chooses one whose load is the lowest by round-robin method or calculating the fastest average response time. This load balancing component is pluggable and enabling a variety of load balancing mechanism to be used. On the other hand, Inter-Grid Executor only forwards the request to the federated service grid where the service is deployed. The access control and endpoint selection are delegated to the federated service grid. In forwarding the request, Inter-Grid Executor has to attach the federation certificate to certify a member of the federation, and replace the requester user ID with one that the operator has as a user of the federated service grid. This is because the federated service grid does not share user authentication information of the requester due to security reasons. Moreover, Inter-Grid Executor inserts the requester user ID into the request header so that the federated service grid can identify the requester and control access from the requester. By separating the service grid that performs user authentication from the different service grid that performs access control, the two service grids do not have to share

users' passwords.

V. Case Study: The Language Grid

Although there are many language resources (both data and programs) on the Internet [3], most intercultural collaboration activities still lack multi-lingual support. To overcome language barriers, we aim to construct a novel language infrastructure to improve accessibility and usability of language resources on the Internet.

To this end, the Language Grid has been proposed [4]. The Language Grid takes a service-oriented collective intelligence approach for sharing language services and creating new services to support their intercultural/multilingual activities by combining language services. In previous works, many efforts were made for combining language resources, such as UIMA [5], GATE [6], and Hart-of-Gold [7]. Their purpose was to analyze a large amount of text data by linguistic processing pipelines. These pipelines consisted of language resources, the most of which were provided as open sources by universities and research institutes. Users could thus collect language resources and freely combine them on those frameworks without considering other stakeholders.

Different from the above frameworks, the purpose of the Language Grid is to support intercultural collaboration by service workflows. A workflow combines language resources associated with complex intellectual property issues, such as machine translators, parallel corpora, and bilingual dictionaries. These resources are provided by service providers who want to protect their ownership, and used by service users who need a part of the resources. Therefore, the Language Grid must coordinate these stakeholders' incentives. That is, it requires language service management to satisfy the following stakeholder's needs as well as language service composition for service users.

In this section, we validate the service grid architecture by using it as basis for constructing the Language Grid. In particular, we focus on language service composition for language service users and federation of the Language Grid.

A. Language Service Domain

Table I shows a language service domain model defined in constructing the Language Grid. The language service domain defines thirteen resource types and sixteen service types. The reason why the number of service types is more than the number of resource types is that the service types contain some composite service types such as BackTranslation, MultihopTranslation, and TranslationWithTemporalDictionary. Moreover, both types have supportedLanguages, supportedLanguagePairs, supporetedLanguagePaths as attributes. They are used to specify languages, images, and audios to be processed by services. Service types also define matchingMethod as an attribute because search function is implemented on language data such as bilingual dictionaries and concept dictionaries.

For example, the resource CaboCha is an instance of DependencyParser type, and has supportedLanguages attribute whose value is Japanese. Service CaboCha, meanwhile, can be both instances of DependencyParse type and MorphologicalAnalysis because dependency parsers generally include morphological analysis results. In addition, service CaboCha belonging to DependencyParse type has four endpoints for load balancing, two of which employ SOAP, and the rest

Table I
Domain Model for Language Services

(a) ResourceType

ResourceType	ResourceTypeAttribute List
BilingualDictionary	supportedLanguagePairs
ConceptDictionary	supportedLanguages
DependencyParser	supportedLanguages
DialogCorpus	supportedLanguages
LanguageIdentification	supportedEncodings, supportedLanguages
MorphologicalAnalyzer	supportedLanguages
ParallelText	supportedLanguagePairs
Paraphraser	supportedLanguages
PictogramDictionary	supportedLanguages, supportedImageTypes
SimilarityCalculator	supportedLanguages
SpeechRecognition	supportedLanguages, supportedAudioTypes, supportedVoiceTypes
TextToSpeech	supportedLanguages, supportedAudioTypes, supportedVoiceTypes
Translator	supportedLanguagePairs

(b) ServiceType

ServiceType	ServiceTypeAttribute List	ServiceInterface[1]
BackTranslation	supportedLanguagePaths	backtranslate
BilingualDictionary	supportedLanguagePairs, matchingMethods	search
ConceptDictionary	supportedLanguages, matchingMethods	searchConcepts, getRelatedConcepts
DependencyParse	supportedLanguages	parseDependency
DialogCorpus	supportedLanguages, matchingMethods	search
LanguageIdentification	supportedEncodings, supportedLanguages	identify
MorphologicalAnalysis	supportedLanguages	analyze
MultihopTranslation	supportedLanguagePaths	multihopTranslate
ParallelText	supportedLanguagePairs, matchingMethods	search
Paraphrase	supportedLanguages	paraphrase
PictogramDictionary	supportedLanguages, matchingMethods, supportedImageTypes	search
SimilarityCalculation	supportedLanguages	calculate
SpeechRecognition	supportedLanguages, supportedAudioTypes, supportedVoiceTypes	recognize
TextToSpeech	supportedLanguages, supportedAudioTypes, supportedVoiceTypes	speak
Translation	supportedLanguagePairs	translate
TranslationWith TemporalDictionary	supportedLanguagePairs	translate

[1]This column describes only operation name due to space limitations

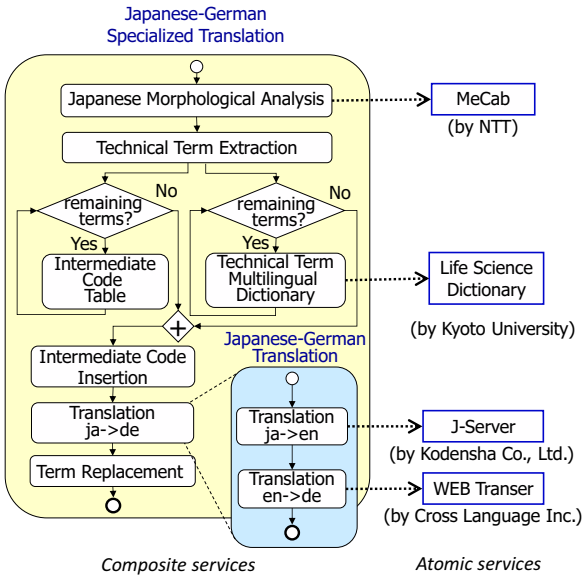

Figure 7. Example of Language Service Composition

of which employ ProtocolBuffers. Every endpoint provides the same interface, whose operation is ”parseDependency”, input is ”language,” and ”sentence,” output is ”analysis result.”

B. Language Service Composition

Among the existing research, Global WordNet Grid [8] is pioneering works on connecting dictionaries in different languages based on word semantics. The Language Grid, however, is a trial to build an infrastructure where users can share and combine various language services.

Fig. 7 shows a domain specialized translation workflow for improving the translation quality of technical sentences. The Language Grid uses Java-coded workflows, and WS-BPEL to describe the workflow. Domain specialized translation workflow consists of several component services: morphological analysis service type, bilingual dictionary service type, and translation service type. To invoke the composite service, service users have to bind a concrete atomic service to each component service type, such as MeCab to the morphological analysis service, Life Science Dictionary to the multilingual dictionary service, and a two-hop translation service consisting of J-Server (machine translator) and WEB-Transer (machine translator) to the translation service. Service users can also invoke other combinations of concrete atomic services by changing the service bindings because resources are wrapped as services with standard interfaces defined by the language service domain model. Moreover, the users can also delay binding services to choose the fastest or most popular one that provides functionality they are interested in.

C. Federated Operation of the Language Grid

We have operated the Language Grid[2] since 2007. The lesson learned from the experience of the operation is importance of the federated operation. Even if we promoted international cooperation, 70-80% of language grid users are from Japan because the operator was located in Japan[1]. Moreover, the most language grid users are occupied by Japanese universities because the operator was also a university. To expand the Language Grid, we have started federated operation of the Language Grid in February 2011 collaboratively with NECTEC in Thai. By this homogeneous federation, we can gather 58 languages and 131 services (51 languages and 110 services from Kyoto Language Grid[2] and 13 languages and 21 services[3] from Bangkok Language Grid).

We also established heterogenous federation between language service domain and education service domain. By reorganizing Tsinghua University’s smart classroom as a set of Web services, we developed a prototype of education service grid. Service composition between the Language Grid and the education service grid realized a multilingual smart classroom where classrooms in multiple countries were connected and multilingual communication was supported [9].

VI. Related Works

The architecture proposed in this work realizes a large-scale and various service compositions by federating multiple service grids.

In grid computing domain, several previous works have proposed methods to federate distributed clusters. When local computation resources are insufficient to meet its users’ requirements, the federation architecture allows usage of other computation resources from the federation [10], [11]. Although these methods reflected our approach, they focus on only connectivity of computation resources, but not services.

On the other hand, in services computing domain, DIRE [12] is one of the framework for the federation of heterogeneous service registries under the lack of a centralized repository like UBR (UDDI Business Registry). This method can increase reusability of services. DIRE is an approach based on the publish and subscribe paradigm. In DIRE, a unique dispatcher provides a physically distributed communication bus to allow registries to publish information about their services, and subscribe to the services they are interested in. This approach can realize efficient dissemination of service information among federated registries, but not compose services across service registries.

[2]http://langrid.org/operation/, http://langrid.org/service_manager/
[1]http://langrid.org/service_manager/users
[2]http://langrid.org/service_manager/language-services
[3]http://www.servicegrid-bangkok.org/langrid-servicemanager/language-services

SOA4All [13], [14] aims to build global service cloud by connecting multiple ESBs and DSBs. However, it does not provide a domain model that organizes services related to a domain and standardizes service interface. As a result, users have to create a process to compose services from the scratch. To form service oriented collective intelligence, it is required to decrease the cost to compose services by organizing services in each domain.

VII. Conclusion

In this paper, we proposed a method that builds service grid for each domain, and federates the service grids to expand service oriented collective intelligence. To this end, we addressed the following two research issues.

Design service domain model

To make general-purpose service grid specific to a certain domain, we constructed a service domain model. This allows service grid operators to define resource type and service type, their attributes, and service interface. As a result, service users can easily change services belonging to the same service type.

Develop service grid federation architecture

We developed service grid federation architecture to realize coordination between homogeneous service grids and heterogeneous service grids. Homogenous service grid federation contributes to increasing the number of services belonging to the same service type. Moreover, heterogeneous service grid federation consisting of different service domains contributes to increasing diversity of services.

To encourage service users and providers to share and compose services, and accelerate the augmentation of collective intelligence, we need not only the proposed service grid federation architecture but also institutional design considering incentives among stakeholders [15].

Acknowledgment

A part of this works was supported by Strategic Information and Communications R&D Promotion Programme from Ministry of Internal Affairs and Communications.

References

[1] Y. Murakami, D. Lin, M. Tanaka, T. Nakaguchi, and T. Ishida, "Service Grid Architecture," in *The Language Grid: Service-Oriented Collective Intelligence for Language Resource Interoperability*, T. Ishida, Ed. Springer, 2011, pp. 19–34.

[2] J. Surowiecki, *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business,Economies, Societies and Nations*. Doubleday, 2004.

[3] K. Choukri, "European Language Resources Association History and Recent Developments," in *SCALLA Working Conference KC 14/20*, 2004.

[4] T. Ishida, "Language Grid: An Infrastructure for Intercultural Collaboration," in *Proc. of the IEEE/IPSJ Symposium on Applications and the Internet (SAINT'06)*, 2006, pp. 96–100.

[5] D. Ferrucci and A. Lally, "UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment," *Journal of Natural Language Engineering*, vol. 10, pp. 327–348, 2004.

[6] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "GATE: An Architecture for Development of Robust HLT Applications," in *Proc. of the Fortieth Annual Meeting on Association for Computational Linguistics (ACL'02)*, 2002, pp. 168–175.

[7] U. Callmeier, A. Eisele, U. Schäfer, and M. Siegel, "The Deep Thought Core Architecture Framework," in *Proc. of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, 2004, pp. 1205–1208.

[8] C. Fellbaum and P. Vossen, "Connecting the Universal to the Specific: Towards the Global Grid," in *Intercultural Collaboration*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2007, no. 4568, pp. 1–16.

[9] Y. Suo, N. Miyata, H. Morikawa, T. Ishida, and Y. Shi, "Open Smart Classroom: Extensible and Scalable Learning System in Smart Space Using Web service Technology," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 6, pp. 814–828, 2009.

[10] R. Ranjan, A. Harwood, and R. Buyya, "A Case for Cooperative and Incentive-Based Federation of Distributed Cluster," *Future Generation Computer Systems*, vol. 24, no. 4, pp. 280–295, 2008.

[11] M. Assuncao, R. Buyya, and S. Venugopal, "InterGrid: A Case for Internetworking Island of Grids," *Journal Concurrency and Computation: Practice & Experience*, vol. 20, no. 8, pp. 997–1024, 2008.

[12] L. Baresi and M. Miraz, "A distributed Approach for the Federation of Heterogeneous Registries," in *Proc. of the International Conference on Service Oriented Computing (ICSOC-06)*, 2006, pp. 240–251.

[13] F. Lecue, R. Gonzalez, Y. Gorronogoitia, M. Radzimski, and M. Villa, "SOA4All: An Innovative Integrated Approach to Services Composition," in *Proc. of the IEEE International Conference on Web Services (ICWS-10)*, 2010, pp. 58–67.

[14] F. Baude, I. Filali, F. Huet, V. Legrand, E. Mathias, P. Merle, C. Ruz, R. Krummenacher, E. Simperi, C. Hammerling, and J. Lorre, "ESB Federation for Large-Scale SOA," in *Proc. of the ACM Symposium on Applied Computing (SAC-10)*, 2010, pp. 2459–2466.

[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," in *Proc. of the First International Conference on Global Interoperability for Language Resources (ICGL'08)*, 2008, pp. 114–12.